Methode zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Systeme

Gerhard Popp

Institut für Informatik
der Technischen Universität München

Methode zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Systeme

Gerhard Popp

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Jürgen Schmidhuber
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Dr. h.c. Manfred Broy
	2.	Univ.-Prof. Dr. Ruth Breu Leopold-Franzens-Universität Innsbruck/ Österreich

Die Dissertation wurde am 26. Januar 2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 04. Juni 2005 angenommen.

Kurzfassung

Ziel der Entwicklung zugriffssicherer Systeme ist es, die dem System anvertrauten Informationen vor unautorisiertem Zugriff zu schützen. Wie Erfahrungen mit der Entwicklung zugriffssicherer Systeme gezeigt haben, reicht es nicht aus, Aspekte der Zugriffssicherheit getrennt nach Fertigstellung des Designs zu betrachten, da dazu oftmals große Teile des Designs überarbeitet werden müssen, sodass dann Zeit- und Kostenpläne nicht eingehalten werden können. Nur zu oft werden die Anforderungen an die Zugriffssicherheit nicht erfüllt, sodass Systeme mit Sicherheitslücken oder Schwachstellen entwickelt werden.

Viele objektorientierte Vorgehensweisen schlagen vor, die Systementwicklung von betrieblichen Informationssystemen mit der Definition der Anforderungen in Form von Anwendungsfällen zu beginnen. Eine vollständige Erfassung und Festlegung der Anforderungen erfordert es, vorab die Abläufe und Informationen des Unternehmens bzw. der Organisation zu modellieren. Die Anforderungen, die das zu erstellende System zu erfüllen hat, können dadurch klar festgelegt werden.

Die vorliegende Arbeit beschreibt insgesamt einen Ausschnitt aus einem Vorgehensmodell auf Basis eines objektorientierten System- und Vorgehensmodells zur Neuentwicklung von zugriffssicheren Informationssystemen. Ein zentrales Ziel des vorgestellten Vorgehensmodells ist die Integration von Aspekten der Zugriffssicherheit in die frühen Phasen der Systementwicklung. Im Vordergrund steht dabei die sukzessive und durchgängige Entwicklung der Anforderungen an die Zugriffssicherheit gemeinsam mit dem Entwurf der Systemfunktionen und des Datenmodells. Die vorgestellte Sicherheitsanalyse wird abstrakt und unabhängig von technischen Details durchgeführt. Durch die Beschreibung in Form generischer Prozessmuster kann die Methode in offene Vorgehensmodelle integriert werden.

Kern dieser Arbeit sind ein Rechtemodell zur Spezifikation von Benutzerrechten und die Integration der Sicherheitsanforderungen in den frühen Phasen der Entwicklung. Das Benutzerrechtemodell basiert auf der Sprache P-MOS zur Navigation über Objekten, Zuständen und Objektbezügen, die sich zur Formulierung von Benutzerrechten im Rahmen des objektorientierten Vorgehens eignet. Im Gegensatz zu existierenden Ansätzen zur Modellierung von Berechtigungen werden die Benutzerrechte außerhalb des Datenmodells in einer Berechtigungsmatrix vorab textuell spezifiziert und im Rahmen der weiteren frühen Entwicklungsstufen in Hinblick auf eine spätere Generierung formalisiert.

Die durchgängige Integration von Sicherheitsanforderungen und die sukzessive Erarbeitung wird in den Entwicklungsphasen der Geschäftsprozessmodellierung, der Anwendungsfallmodellierung und der Analyse diskutiert. Die einzelnen Arbeitsschritte zur Modellierung der Sicherheit in Verbindung mit der Entwicklung der Systemfunktionalität werden anhand von Prozessmustern beschrieben. Im Rahmen der Entwicklung sind bestehende Produktartefakte mit Informationen zur Abdeckung der Zugriffssicherheit anzureichern sowie neue Produktartefakte hinzuzufügen.

Die vorgestellte Analyse von Aspekten der Zugriffssicherheit basiert auf der Integration von Schutzzielen innerhalb von Abläufen und Datenmodellen. Im Vordergrund stehen dabei die Schutzziele Vertraulichkeit, Integrität, Verbindlichkeit sowie Authentizität. Aufbauend auf diesen Schutzzielannotationen werden Bedrohungen, Risiken und Maßnahmen innerhalb eines Sicherheitsmikroprozesses ermittelt und die Benutzerrechte formalisiert.

Da die Beschreibungssprache UML (Unified Modeling Language) im Rahmen von objektorientierten Vorgehensmethoden in großem Maße angewendet wird, aber mit ihren grafischen, textuellen und formalen Beschreibungen keine geeigneten Notationen zur Modellierung von Aspekten der Zugriffssicherheit innerhalb den frühen Phasen einer Systementwicklung bietet, werden notwendige Erweiterungen der Beschreibungstechniken vorgestellt. Insbesondere werden Stereotypen zur Annotation von Schutzzielen in Aktivitäts- und Klassendiagrammen eingeführt.

Die erarbeiteten Konzepte der Vorgehensbausteine, des Rechtemodells und der erweiterten grafischen Beschreibungstechniken werden prototypisch an einer Fallstudie zur Projektverwaltung untersucht.

Dankesworte

Vorab möchte ich mich bei allen bedanken, die mich in den letzten Jahren unterstützt und mir so das Gelingen dieser Arbeit ermöglicht haben.

Ein besonderer Dank geht an Herrn Prof. Dr. Dr. h. c. Manfred Broy, der es mir ermöglichte, an diesem aktuellen Thema zu arbeiten. Die wissenschaftliche Betreuung, die großen Freiräume und das entgegengebrachte Vertrauen gebühren einen besonderen Dank. Ein weiterer besonderer Dank ergeht an Frau Prof. Dr. Ruth Breu für die zahlreichen inhaltlichen Diskussionen sowie für die Übernahme des Zweitgutachtens.

Großer Dank gebührt auch den Kollegen des MEwaDis-Projekts, Herrn Martin Deubler, Herrn Johannes Grünbauer und Herrn Guido Wimmel, die interessante Impulse für meine Arbeit lieferten, mir sehr viel Freiraum im Projekt einräumten und für ein angenehmes Arbeitsklima sorgten.

Des Weiteren danke ich den Innsbrucker Kollegen, Frau Joanna Chimiak-Opoka, Herrn Klaus Burger und Herrn Michael Hafner, für die wertvollen Diskussionen und die gute thematische Zusammenarbeit.

Für das aufmerksame Durchlesen und die zahlreichen Kommentare zur endgültigen Fassung meiner Arbeit möchte ich mich bei Herrn Martin Deubler, Herrn Johannes Grünbauer, Herrn Thomas Kuhn und Frau Dr. Katharina Spies besonders bedanken.

Nicht zuletzt danke ich meinen Eltern Anna und Robert sowie meiner Freundin Heidi für die Geduld und Rücksicht sowie für den seelischen und moralischen Beistand.

Inhaltsverzeichnis

1 Einführung

Die Durchdringung unseres Lebens mit Rechnern ist unübersehbar. Während anfangs nur große Unternehmen und Behörden, wie beispielsweise Banken und Militär, ihre Aufgaben rechnergestützt bearbeitet haben, fand vor allem innerhalb der letzten zehn bis zwanzig Jahre eine Durchdringung des privaten Bereichs statt. Die Wiedereinführung von Rabatten auf Basis von elektronischen Kundenkarten, der Entertainment-Bereich in Form von Satellitenanlagen oder Video-on-Demand sowie das mittlerweile klassische Online-Banking sind neben dem Informationsangebot und der Bestellabwicklung im Internet weit verbreitete, rechnergestützte Dienste, die von vielen Privatpersonen genutzt werden.

Die Vernetzung in Form des Internets, von lokalen Firmennetzen sowie von drahtlosen Verbindungen im LAN- und WAN-Bereich, stellt die Basis für den notwendigen Informationsaustausch dar. So werden unter anderem bei Drogeriehandelsketten die Einkäufe in den Filialen zu einem zentralen Rechner für die Rabattberechnung übermittelt oder Kontobuchungen in Form von Last- und Gutschriften übertragen. Das Navigationssystem im Auto informiert den Fahrer über die aktuelle Position und den weiteren Streckenverlauf, ein Flug kann mittels eines Web-Formulars gebucht werden und Abrechnungsdaten vom Arzt sind den Krankenkassen elektronisch zu übermitteln.

Bei den von Rechnern zu verarbeitenden Daten handelt es sich oftmals um wertvolle Informationen. Durch kriminelle Vorgehensweisen wird versucht, diese Daten auszuspionieren oder zu manipulieren. Berichte und Statistiken (wie beispielsweise [Sil04]) zeigen, dass die Computerkriminalität in den letzten Jahren stark angestiegen ist. Die Bandbreite hierbei reicht von kleinen Angriffen, wie etwa das Ausspionieren von Personaldaten eines Mitschülers bis hin zu terroristischen Übergriffen, die ausgeführt werden können, ohne ein zu attackierendes Land betreten und ohne Security-Checks am Flughafen über sich ergehen lassen zu müssen.

Da in den vergangenen Jahrzehnten größtenteils Einzelplatzrechner oder isolierte Netzwerke eingesetzt wurden, reichte es zumeist aus, die Rechner in abgesperrten Räumen vor mutwilligen Angriffen zu schützen oder die Rechner mit einer einfachen Login-Maske durch Benutzerkennwort und Passwort zu schützen. Die Zugriffssicherheit konnte hierbei mit einfachen Mitteln gewährleistet werden.

Durch die zunehmende Vernetzung, Verteiltheit und Mobilität der Systeme während der letzten Jahre ergeben sich völlig neue Angriffsmöglichkeiten auf einzelne Rechner, spezielle Anwendungen oder ganze Rechnernetze. Beispielsweise können Angreifer durch die Anbindung der Rechner an öffentliche Netze versuchen, die über das Netz ausgetauschten Informationen mitzulesen oder direkt auf die Rechner zu gelangen, um Daten auszuspionieren oder um illegal Applikationen auf fremden Rechnern auszuführen. Um derartige und weitere Angriffe auf Rechner zu verhindern, ist es notwendig, zugriffssichere Systeme zu entwickeln, die Angriffe sowohl auf die Rechner selbst sowie auch auf die zu übertragenden Daten erfolgreich abwehren können.

Die Softwarekrise in den 60er Jahren des letzten Jahrhunderts (siehe [Bal96]) hat gezeigt, dass die Entwicklung softwareintensiver Produkte schwierig ist und dass hierzu konkrete Vorgehensweisen zur Strukturierung und Beschreibung angewendet werden müssen. Da für die Entwicklung von zugriffssicheren Systemen zusätzlich Aspekte der Zugriffssicherheit betrachtet werden müssen, steigt mit der Komplexität der Systemanforderungen auch die Schwierigkeit der Systementwicklung und die Fehleranfälligkeit (vgl. [Jür04]) der Systeme. Die Notwendigkeit von so genannten Entwicklungstechniken wird hierbei verstärkt gefordert.

Bei der Entwicklung zugriffssicherer Systeme wurde in den vergangenen Jahren die Notwendigkeit von eigenen Vorgehensmodellen erkannt [Eck03]. Hierzu entstanden erste Vorgehensmodelle, die größtenteils eine Erweiterung bestehender Entwicklungsmethoden sind. Diese Entwicklungsmethoden haben sich jedoch nicht etabliert, sodass Aspekte der Zugriffssicherheit in einer Vielzahl von Systemen ohne Systematik entwickelt werden. Gründe für das Scheitern sind beispielsweise, dass diese ersten Ansätze nur eine Annäherung an ein Vorgehensmodell waren oder dass sie fest an bestehende Vorgehensmodelle gebunden sind, wie beispielsweise an das Wasserfallmodell in [Eck03] oder an das V-Modell in [GDB]. Die Entwicklung von Aspekten der Zugriffssicherheit kann dabei nur dann durchgeführt werden, wenn die Systeme nach diesen Vorgehensmodellen entwickelt werden. Auch wird in diesen ersten Ansätzen zur Entwicklung zugriffssicherer Systeme [Eck03, GDB, Vet01, AW03b, VM02, And01] auf aktuelle Vorgehensweisen, wie beispielsweise die objektorientierte Modellierung, nicht Bezug genommen; sie beschränken sich auf die klassischen phasenstrukturierten Ansätze [PB96] zur Entwicklung zugriffssicherer Systeme. Letztendlich wird auch auf die iterative Softwareentwicklung, wie sie heute für große Systeme üblich ist, kaum eingegangen. Derzeit fehlt es an einer Weiterentwicklung dieser ersten Konzepte und an der Entwicklung einer integrierten Methode zur Entwicklung sicherer Systeme.

Bei der Entwicklung sicherer Systeme ohne dediziertes Vorgehensmodell wird meist versucht, nach der Umsetzung der funktionalen Anforderungen die Systeme gegenüber Angriffen abzusichern, indem Aspekte der Zugriffssicherheit als Erweiterung angefügt werden. Da jedoch bei der Entwicklung zugriffssicherer Systeme ein Großteil der Abläufe und Daten an die Techniken zur Gewährleistung der Sicherheit angepasst werden müssen, ist es nicht möglich, die Aspekte der Zugriffssicherheit am Ende der Produktentwicklung anzufügen. Techniken zur Realisierung der Zugriffssicherheit können eben nicht als isolierter Baustein angefügt werden. Alle Abläufe und Daten müssen auf Sicherheitsaspekte überprüft und gegebenenfalls angepasst werden. Ein großes Problem stellt dabei das nahende Ende der Entwicklung dar, da gegen Ende zumeist Zeit und Budget schon überzogen sind und die Integration der Sicherheitsaspekte somit nicht mehr im erforderlichen Ausmaß verfolgt werden kann.

1.1 Zielsetzung

Sicherheit spielt in heutigen Systemen mehr und mehr eine entscheidende Rolle. Derzeit fehlt ein methodisches Vorgehen zur Entwicklung zugriffssicherer Systeme.

Im Rahmen dieser Arbeit wird ein Vorgehen entwickelt, welches zeigt, wie eine schrittweise Entwicklung der Zugriffssicherheit in den frühen Phasen der Softwareentwicklung betrieblicher Informationssysteme integriert werden kann. Im Vordergrund steht dabei die Entwicklung der Anforderungen an die Zugriffssicherheit. Neben der Analyse des Vorgehensmodells werden

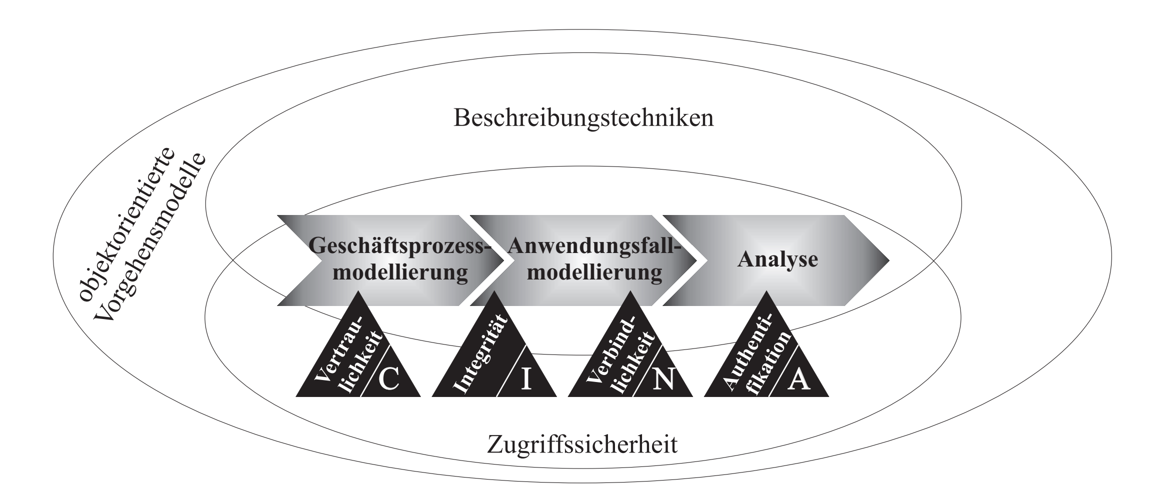


Abbildung 1.1: Einflussfaktoren für die Entwicklung zugriffssicherer Systeme

grafische Beschreibungstechniken im Kontext der Unified Modeling Language (UML) und für zugriffssichere Systeme eine prädikative Spezifikation von Benutzerrechten auf Basis eines rollenbasierten Benutzerrechtemodells vorgestellt.

Das Vorgehensmodell zur Entwicklung zugriffssicherer Informationssysteme wird von den in Abbildung 1.1 dargestellten Faktoren beeinflusst. Basis der Untersuchungen sind die drei Prozessabschnitte Geschäftsprozessmodellierung, Anwendungsfallmodellierung und Analyse, wie sie in neueren objektorientierten Ansätzen vorzufinden sind. Dort gilt es, Anforderungen an die Zugriffssicherheit zu integrieren und geeignete Beschreibungstechniken zu untersuchen. Im Vordergrund stehen dabei die Auswirkungen der Aspekte Vertraulichkeit, Integrität, Verbindlichkeit und Authentifikation der Zugriffssicherheit auf die genannten Prozessabschnitte.

Ein *Vorgehensmodell* regelt den Ablauf des Lösungsprozesses und unterteilt ihn in überschaubare Abschnitte zur Planung, Durchführung, Entscheidung und Kontrolle (vgl. [PB96]). Ein Kernziel der Arbeit ist es, für die Entwicklung zugriffssicherer Software einen Lösungsprozess der Anforderungsspezifikation bereitzustellen und zu zeigen, wie die Lösung in die zeitliche Abfolge der Phasen, den so genannten Softwarelebenszyklus, eingebettet werden kann.

Seit Anfang der 80er Jahre werden in der Softwareentwicklung mehr und mehr objektorientierte Methoden angewendet, die vor allem die Ziele der Erweiterbarkeit, der Wiederverwendung und der Wartbarkeit unterstützen [Bre01]. Besonders bei der Entwicklung von großen Programmen eignen sich diese Methoden durch ihre iterativen Entwicklungsansätze und neuartigen Spezifikationstechniken von Anforderungen beispielsweise in Form von Anwendungsfällen. In der vorliegenden Arbeit wird gezeigt, wie eine objektorientierte Vorgehensmethode bei der Definition der Anforderungen von zugriffssicheren Systemen erfolgreich eingesetzt werden kann, wie die Sicherheitsanforderungen verfeinert und wie Iterationen dabei nutzbringend eingeführt werden können.

Die Zugriffssicherheit beschäftigt sich mit der Abwehr von absichtlichen Angriffen. Zur genauen Definition dieses Begriffes benötigen wir zunächst den Begriff der Funktionssicherheit (vgl. [Eck03]): Unter der Funktionssicherheit (engl. *Safety*) eines Systems verstehen wir die Eigenschaft, dass die realisierte Ist-Funktionalität der Komponenten mit der spezifizierten Soll-Funktionalität übereinstimmt. Die Zugriffssicherheit ist die Eigenschaft eines funktions-

sicheren Systems, nur solche Systemzustände anzunehmen, die zu keinem unautorisierten Zugriff auf Systemressourcen und Kommunikationswege und dabei insbesondere auf Daten führen. Im Gegensatz zur Datensicherheit (engl. *Protection*) bleiben bei der Zugriffssicherheit die Datensicherung sowie der Datenschutz außen vor.

Wir beschäftigen uns im Rahmen der frühen Softwareentwicklung damit, unbefugte Zugriffe oder Modifikationen von Daten und Informationen, die innerhalb der Software gespeichert oder übertragen werden, zu verhindern. Die vorgestellte Definition der Zugriffssicherheit erfordert die Funktionssicherheit als Voraussetzung der Zugriffssicherheit. Im Rahmen dieser Arbeit beschränken wir uns daher bei unseren Überlegungen auf den Aspekt der Zugriffssicherheit. Wir zeigen, wie ein System gegen unerwünschte Zugriffe von außen zu sichern ist. Die Funktionssicherheit sehen wir dabei als gegeben an.

Sicherheitsanforderungen werden in existierenden Methoden [JBR99, Kru00, Bal98] zur Softwareentwicklung zumeist im Rahmen der Ermittlung der *nichtfunktionalen Anforderungen* lose neben den funktionalen Anforderungen betrachtet. Sie werden oftmals auch als Software-Qualitätsmerkmale [Bal98] oder als Non-Behavioral Aspects [Boo94] bezeichnet. Ziel des vorgestellten Ansatzes ist es, Anforderungen an die Zugriffssicherheit bei der Beschreibung der funktionalen Anforderungen zu integrieren und schrittweise zu verfeinern. Am Ende der Anforderungsspezifikation können damit die Anforderungen der Zugriffssicherheit in die funktionalen Abläufe und in die Struktur des zu entwickelnden Systems integriert werden.

Der im Rahmen dieser Arbeit vorgestellte Ansatz zur Verfeinerung der Zugriffssicherheit basiert auf der möglichen Aufteilung der Anforderungen an die Zugriffssicherheit in die Schutzziele [Eck03] Vertraulichkeit, Integrität, Verbindlichkeit, Authentifikation und Verfügbarkeit. Da diese Aufteilung als erster Schritt zur Verfeinerung durchzuführen ist, kann der vorgestellte Ansatz nicht auf andere nichtfunktionale Anforderungen, wie zum Beispiel Effizienz, Änderbarkeit oder Übertragbarkeit, übertragen werden. Diese weiteren nichtfunktionalen Anforderungen lassen sich nicht in die gegebenen Unteranforderungen aufteilen.

Mit den Vorgehensprozessen eng verbunden sind Produktartefakte, welche die Ein- und Ausgaben zu einzelnen Prozessschritten darstellen. Neben der Prozessanalyse stehen die Produktartefakte zur Spezifikation der Anforderungen an die Zugriffssicherheit ebenfalls im Vordergrund unserer Betrachtungen. Hierzu sind bestehende Produktartefakte aus der objektorientierten Softwareentwicklung zu erweitern und zusätzliche Produktartefakte zu integrieren. Zur Dokumentation benötigen wir geeignete *Beschreibungstechniken*, die es uns erlauben, Aspekte der Zugriffssicherheit in bestehenden Diagrammen zu annotieren und in eigenen Produktartefakten zu beschreiben. Da die Unified Modeling Language (UML) [OMG03, FS98, BRJ98] die objektorientierten Softwareentwicklungsmethoden durchdrungen hat und unser Vorgehen auf objektorientierte Vorgehensweisen ausgerichtet ist, wird eine Erweiterung von UML-Diagrammen zur Integration von Aspekten der Zugriffssicherheit vorgestellt. Für die Ablaufmodellierung sind neue Techniken gefordert, die wir mittels des Erweiterungsmechanismus der UML in Diagramme integrieren. Neben Erweiterungen der UML benötigen wir auch textuelle Beschreibungen, beispielsweise zur Spezifikation von Anforderungen der Zugriffssicherheit in den Anwendungsfällen oder zur Beschreibung von Bedrohungen und Risiken.

Neben einer Modellierung der Schutzziele in Struktur und Verhalten sowie von Bedrohungen und Risiken stellt die *Zugriffskontrolle* als Schutz von Daten und Systemressourcen ein zentrales Thema bei der Entwicklung zugriffssicherer Software dar. In der Literatur wurde jedoch die Modellierung von Benutzerrechten in frühen Phasen der Softwareentwicklung betrieblicher

Informationssysteme kaum betrachtet, obwohl eine frühzeitige Modellierung schon bereits ab der Geschäftsprozessmodellierung sinnvoll ist, da zu diesem Zeitpunkt Anforderungsentwickler, Auftragnehmer, Auftraggeber und Anwender eng zusammenarbeiten. Bereits hier können Benutzerrechte beispielsweise nach dem rollenbasierten Paradigma (vgl. *Role Based Access Control (RBAC)* in [FCK03, FSG+01, San96]) textuell spezifiziert werden. Benutzerrechte werden dabei an abstrakte Rollen gebunden und jeder Benutzer ist an eine oder mehrere Rollen gebunden. Den Rollen werden eine Menge von Berechtigungen zugeordnet, wobei jede Berechtigung die Art des Zugriffs festlegt. Die somit ermittelten Benutzerrechte können innerhalb der Anwendungsfallmodellierung und der Analyse schrittweise ausgebaut und verfeinert werden. Für eine spätere Codegenerierung von Berechtigungen auf Methoden sind diese textuellen Benutzerrechte in kurze und prägnante Prädikate erster Ordnung und als OCL-Ausdrücke zu transformieren.

Mit der Ausweitung der Vernetzung und dem Fortschritt der Technik werden in Zukunft auch bereits bestehende Softwarelösungen mit Aspekten der Zugriffssicherheit erweitert werden oder durch neue Software abzulösen sein. Hierzu sind Vorgehensweisen zur Analyse von bestehenden Sicherheitseigenschaften notwendig. Im Rahmen dieser Arbeit beschränken wir uns jedoch auf die Neuentwicklung zugriffssicherer Informationssysteme. Im Rahmen von Prozessbetrachtungen treten neben dem eigentlichen Vorgehensmodell auch Managementaufgaben und Organisationsaspekte für eine Integration der Prozesse innerhalb von Institutionen und Unternehmen auf. Wir beschränken uns hier jedoch auf das Vorgehen mit den notwendigen Produktartefakten und Beschreibungstechniken.

1.2 Ergebnisse

Die Ergebnisse stellen grundlegende Konzepte und Ansätze bereit, die notwendig sind, um Sicherheitsanforderungen bei der Entwicklung von zugriffssicheren Systemen durchgängig, sukzessive und iterativ zu entwickeln.

Im Rahmen dieser Arbeit wird ein Ausschnitt aus einem Vorgehensmodell auf der Grundlage eines objektorientierten System- und Vorgehensmodells zur Neuentwicklung von zugriffssicheren Informationssystemen beschrieben. Basierend auf einem ausführlichen Literaturvergleich von bestehenden Methoden zur Entwicklung sicherer Systeme werden Prozessanforderungen abgeleitet.

Ausgehend von einer Projektidee und einem übergeordneten, abstrakten Sicherheitsziel werden in der Geschäftsprozessmodellierung, in der Anwendungsfallmodellierung und in der Analyse Aspekte der Zugriffssicherheit durchgängig, sukzessiv und iterativ erarbeitet. Es wird gezeigt, wie die Modellierung der Zugriffssicherheit in bestehende Modellierungsaktivitäten einzubetten ist und welche Ein- und Ausgaben in Form von Produktartefakten, insbesondere für die sukzessive Modellierung der Sicherheit, in den einzelnen Teilschritten notwendig sind.

Besondere Betrachtung findet die Beschreibung der Benutzerrechte. Zu Beginn werden diese innerhalb der Geschäftsprozessmodellierung für Daten und Aktivitäten textuell in einer Matrix beschrieben und in der Anwendungsfallmodellierung sowie der Analyse verfeinert und ergänzt. Diese textuell spezifizierten Zugriffsbeschränkungen werden im Laufe der Anforderungsspezifikation formalisiert. Mit dieser implementierungsunabhängigen Beschreibung, bei

der das Domänenmodell sowie die Ablaufbeschreibungen frei von Benutzerrechtsspezifikationen sind, wird die Komplexität dieser Beschreibungen gering gehalten. Weiterhin können aus den formal spezifizierten Benutzerrechten später bei der Implementierung automatisch Zugriffsprozeduren generiert werden, sodass vor der Ausführung von Methoden die Zugriffsrechte des ausführenden Benutzers überprüft werden können.

Die implementierungsunabhängige, prädikative Spezifikation der Benutzerrechte basiert auf dem Spezifikationsrahmenwerk P-MOS (vgl. [Bre01]). Basierend auf dieser Sprache von Ausdrücken und Prädikaten zum Navigieren über Objekte, Zustände und Objektbezüge wird ein akteurzentriertes Modell zur Spezifikation von Benutzerrechten eingeführt. Grundidee dieses Modells ist die Erweiterung von Methodenaufrufen um eine Berechtigungsspezifikation. Wie eine Vorbedingung muss die Berechtigung vor dem Ausführen der Methode überprüft werden. Die Ausdrücke und Prädikate in P-MOS beziehen sich dabei auf das Domänenmodell des zu entwickelnden Systems, sind rein statisch und getypt. Berechtigungsüberprüfungen liefern damit einen Wert zurück, sind aber nicht zustandsverändernd. Bei der detaillierten Untersuchung der Benutzerrechte werden diese in Zugriffs- und Ausführungsrechte eingeteilt und es wird ein Berechnungsverfahren angegeben, wie aus Zugriffsberechtigungen eine Ausführungsberechtigung ermittelt werden kann. Zudem werden verschiedene Typen von Ausführungsrechten eingeführt.

Basierend auf verschiedenen neueren objektorientierten Methoden zur Systementwicklung betrieblicher Informationssysteme wird die durchgängige und sukzessive Erarbeitung von Aspekten der Zugriffssicherheit in den Phasen der Geschäftsprozessmodellierung, der Anwendungsfallmodellierung und der Analyse gezeigt. Zur Spezifikation der Anforderungen in diesen frühen Phasen werden UML-Diagramme verwendet, zur Darstellung der Anforderungen der Zugriffssicherheit werden diese um notwendige Konstrukte ergänzt. Mittels des Erweiterungsmechanismus der UML werden Stereotypen für die Schutzziele Vertraulichkeit, Integrität, Verbindlichkeit und Authentifikation in Klassen- und Aktivitätsdiagrammen eingeführt. Zur Darstellung von Sitzungen werden Aktivitätsdiagramme um spezielle Assoziationen ergänzt. Die Erweiterung des Produktmodells um eigene sicherheitsspezifische Produkte sowie die Erweiterung von Standardprodukten gegenüber herkömmlichen Produktmodellen wird gezeigt. Die Prozessbausteine für die Modellierung der Zugriffssicherheit in den frühen Phasen der Softwareentwicklung werden mit Hilfe von Prozessmustern [Amb98, Amb99, GMP+03b] dargestellt. Damit lassen sich die Prozesse zur Sicherheitsanalyse auch in neue generische Prozessmodelle integrieren.

Besondere Bedeutung wird auch der iterativen Entwicklung der Anforderungen an die Zugriffssicherheit eingeräumt, wie diese in den objektorientierten Methoden gefordert wird. Besonders für die Entwicklung großer Systeme werden in allen drei Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme Iterationsmöglichkeiten aufgezeigt. Hierzu wird ein iterativ anwendbarer Sicherheitsmikroprozess zur Ermittlung von Bedrohungen, Risiken und Maßnahmen vorgestellt.

Alle erarbeiteten Konzepte werden an einer Fallstudie zur Projektverwaltung untersucht.

Der in dieser Arbeit vorgestellte Ansatz zur Modellierung von Aspekten der Zugriffssicherheit in den frühen Phasen der Softwareentwicklung reduziert gegenüber anderen Ansätzen die Anzahl von Objekten im Domänenmodell, indem die Berechtigungen außerhalb in einer Benutzerrechtematrix spezifiziert werden. Während in den Ansätzen [LBD02, BDL03, BKL01, KPP03] für jedes Objekt, für das der Zugriff beschränkt werden muss, ein Objekt zur Spezifikation der

Berechtigung eingesetzt wird, verdoppelt sich in diesen Ansätzen die Zahl der Objekte und Assoziationen nahezu. Durch die prädikative Spezifikation der Benutzerrechte auf Basis von Methodenberechtigungen können die Benutzerrechte im Rahmen der Implementierung aus der formalen Beschreibung in ausführbaren Code zur Abprüfung der Berechtigungen generiert werden. Weiterhin wird durch die vorgestellte Methode die durchgängige Entwicklung von Aspekten der Zugriffssicherheit, wie sie bei der Entwicklung von funktionalen Aspekten in verschiedensten Ansätzen diskutiert wird (beispielsweise in [JBR99, Kru00, Bre01, DW98, IAB]) auf den Bereich der Zugriffssicherheit übertragen. Anstelle von textuellen Anforderungsbeschreibungen in Form von nichtfunktionalen Anforderungen, für die in den vorhandenen Ansätzen kaum konkrete Schritte zur Umsetzung angegeben sind, wird im Prozess die schrittweise und aufbauende Erstellung von Produkten zur Spezifikation der Sicherheitsanforderungen gezeigt. Die vorgestellte Methode ist insbesondere deshalb an objektorientierte Verfahren angelehnt, da diese Verfahren neben den Prozessen die Produktartefakte als Ein- und Ausgaben zu den Prozessen betrachten und da für die Dokumentation die grafische Beschreibungssprache UML verwendet wird. Der Einsatz der UML eignet sich besonders deshalb für die vorgestellte Methode, da die UML durch ihre Erweiterungsmöglichkeiten [OMG03, EP00, BRJ98] Annotationen von Aspekten der Zugriffssicherheit ermöglicht, die UML zur Beschreibung von Geschäftsprozessen verwendet wird [EP00], die UML zudem leicht anwendbar und in der Praxis sowie in der Wissenschaft als semiformale Beschreibungssprache Anwendung findet. Die Erarbeitung von Aspekten der Zugriffssicherheit innerhalb der Geschäftsprozessmodellierung ist insbesondere deshalb sinnvoll, da dadurch das übergreifende Zusammenspiel von verschiedenen Tätigkeiten innerhalb der komplexen Prozesslandschaft von Organisationseinheiten verbessert wird und dabei gemeinsam mit Auftraggebern, Anwendern, Auftragnehmern und Anforderungsentwicklern Aspekte der Zugriffssicherheit in die Geschäftsprozesse aufgenommen werden können.

1.3 Aufbau der Arbeit

Kapitel 2 stellt die Grundlagen zu Vorgehensmodellen dar und definiert grundlegende Begriffe zur Sicherheit.

Um die abstrakt eingeführten Bausteine der Methode, die Benutzerrechtemodellierung, die Beschreibungstechniken und die Produktartefakte, an einem durchgängigen Beispiel zu illustrieren, wird in Kapitel 3 als Fallstudie das Projektverwaltungssystem *TimeTool* vorgestellt.

Das akteurzentrierte Modell zur Spezifikation von prädikativen Benutzerrechten wird in Kapitel 4 vorgestellt. Neben einer allgemeinen Vorstellung des RBAC-Ansatzes wird das zugrunde liegende Spezifikationsframework P-MOS vorgestellt und die formale Modellierung von Benutzerrechten eingeführt. Dabei wird auch die Spezifikation von Zugriffsrechten in OCL sowie eine Erweiterung des methodenbasierten Ansatzes auf Klassen gezeigt.

Kapitel 5 stellt den Prozess zur Entwicklung einer Anforderungsspezifikation zugriffssicherer Systeme vor. Hierbei werden vorab existierende Vorgehensansätze untersucht und daraus Anforderungen an ein Vorgehen zur Entwicklung zugriffssicherer Systeme abgeleitet. Drei Prozessmuster, zur Sicherheitsanforderungsdefinition, zur Integration der Anforderungsspezifikation in den Softwarelebenszyklus und zum Sicherheitsmikroprozess, führen das eigentliche Entwicklungsvorgehen ein. Ein Überblick über die Produktartefakte und eine Bewertung der Prozessanforderungen für den eingeführten Prozess schließen das Kapitel ab.

Das Vorgehen der Geschäftsprozessmodellierung, der Anwendungsfallmodellierung und der Analyse zugriffssicherer Systeme wird in Kapitel 6 bis Kapitel 8 beschrieben. Innerhalb der einzelnen Teilphasen gehen wir auf die Berechtigungsmodellierung, die Beschreibungstechniken, die Produktartefakte sowie auf die Prozesse ein. Zur Verdeutlichung werden die Konzepte hier mit Beispielen aus der Fallstudie *TimeTool* erläutert.

In Kapitel 9 sind die erzielten Ergebnisse kritisch bewertet und zusammengefasst. Ein Ausblick schließt die Arbeit ab.

1.4 Verwandte Arbeiten

Erste, einfach handhabbare Vorgehensweisen zur Entwicklung sicherer Systeme sind in [Eck03, VM02, GDB, Vet01, VWW02] zu finden. Diese Ansätze stellen eine Erweiterung von Standardvorgehensmodellen, wie beispielsweise das Wasserfallmodell oder das Spiralmodell, um Sicherheitsaspekte dar. Eine genauere Betrachtung der Sicherheitsanalyse ist in [Eck03] zu finden, wobei hier Funktionalitätsanforderungen und Sicherheitsaspekte größtenteils losgelöst voneinander entwickelt werden. [VM02, GDB] stellen nur Grundsätze für eine sichere Entwicklung und die Einbettung in Vorgehensmodelle dar, eine klare Durchdringung der Entwicklung wird hierbei nicht gezeigt. In [Vet01] wird ein erster Ansatz für eine integrierte Entwicklung von Sicherheit und Funktionalität mit dem Ziel der Evaluierung vorgestellt. Die Sicherheit wird hierbei während der gesamten Entwicklung betrachtet, jedoch basiert dieses Verfahren auf dem linearen Wasserfallmodell. Ross Anderson behandelt Security Engineering in [And01]. Im Vordergrund stehen dabei jedoch mehr die Techniken, die in sicheren Systemen bei der Entwicklung Einsatz finden. Der Autor beschäftigt sich in geringem Maße mit top-down orientierten und iterativen Vorgehensweisen. All diese Ansätze bilden die Basis für die Untersuchungen zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Systeme, insbesondere werden sie zur Definition der Prozessanforderungen herangezogen. [GHS03] betrachtet Sicherheitseigenschaften in den verschiedenen Phasen der Softwareentwicklung, gibt aber wenige methodische Richtlinien zur durchgängigen und sukzessiven Entwicklung. [Bas93] betrachtet die Evolution von Informationssystemen und vergleicht verschiedene Methoden zur Analyse von Sicherheitseigenschaften. Die darin betrachteten Methoden basieren jedoch auf eingeschränkten Checklisten.

Basis für eine strukturierte, durchgehende und sukzessive Betrachtung von Aspekten der Zugriffssicherheit ist die Unterteilung der Anforderungen in Schutzziele, wie dies beispielsweise in [Eck03, BSW01, Dos01] vorgestellt wird.

Anwendungsfälle zur Sicherheit wurden bereits mehrfach diskutiert. In [FH97b] werden Überlegungen zu einer Erweiterung von Anwendungsfällen auf nichtfunktionale Eigenschaften, darunter auch Sicherheit, gestellt. Diese, auf einem Fragenkatalog basierende Analyse liefert jedoch nur sehr abstrakte Anforderungen und ein durchgängiges Verfahren zur Entwicklung von Anwendungsfällen mit Sicherheitsaspekten geht daraus nicht hervor. Das Konzept der *Misuse Cases* [SO01] (auch als *Abuse Cases* bezeichnet) beschäftigt sich mit der Analyse und Spezifikation von Bedrohungen, während sich Sicherheitsanwendungsfälle (engl. *Security Use Cases*) mit der Spezifikation von Sicherheitsanforderungen beschäftigen. [Fir03] zeigt einen Vergleich der beiden Konzepte und stellt sehr grob granulare Sicherheitsanwendungsfälle vor. Das Konzept der *Security Use Cases* wird in dieser Arbeit bei der Anwendungsfallmodellierung zugriffssicherer Systeme für Sicherheitsgrundfunktionen übernommen. Da es schwierig

ist, *Misuse Cases* vollständig zu erfassen und hierzu keine methodischen Richtlinien angegeben werden, betrachten wir dieses Konzept nicht weiter.

Die Modellierung von Anforderungen an die Autorisierung in Sicherheitsanwendungsfällen wird in [AW03a] diskutiert. Dieser Ansatz beschäftigt sich dabei insbesondere mit der Vermeidung von Inkonsistenzen und Unvollständigkeiten. Eine frühe Modellierung von Sicherheitseigenschaften wird auch in [AW03b] vorgeschlagen. Dabei wird das Konzept der Anwendungsfälle bei der Modellierung der Anforderungen diskutiert und eine erweiterte Anwendungsfallbeschreibung vorgestellt. In Verbindung mit einer Erweiterung der UML um Sicherheitsaspekte (UMLsec) wurden Sicherheitsanwendungsfälle in [JPW03, PBJW03] erörtert. In [BBH+03, BBHP04, BBHP05] wird eine erste Integration von Sicherheitsanwendungsfällen in ein dediziertes Prozessmodell für die Entwicklung sicherer Systeme vorgestellt. Diese Arbeiten von [AW03a, AW03b, JPW03, PBJW03, BBH+03, BBHP04, BBHP05] stellen wichtige Grundlagen und Konzepte für die Betrachtungen zur Anwendungsfallmodellierung zugriffssicherer Systeme in dieser Arbeit dar, wie beispielsweise die erweiterte strukturelle Beschreibung von Anwendungsfällen und die Integration von Sicherheitsanwendungsfällen zu Sicherheitsgrundfunktionen.

Im Kontext einer formalen Fundierung von Anforderungsspezifikationen zur Sicherheit und Protokollverifikationen sind eine Reihe von Arbeiten entstanden (siehe beispielsweise [Jür04, GSG99, vOL02, JW03]). Der in dieser Arbeit vorgestellte Ansatz zur formalen Spezifikation von Benutzerrechten basiert auf dem RBAC-Modell [FSG+01, FCK03, San96, Eck03] und auf der formalen Sprache P-MOS [Bre01, Bre04]. Ein Schema zur Modellierung von Benutzerrechten im Kontext von Anwendungsfällen wurde in [FH97a] erstmals untersucht. Unser Ansatz basiert auf einigen dieser Ideen, stellt jedoch eine grundlegendere Theorie zur Verfügung. Die in dieser Arbeit vorgestellte akteurbezogene Modellierung von Benutzerrechten grenzt sich von [LBD02, BDL03] dahingehend ab, dass die Benutzerrechte schon in einer frühen Phase der Softwareentwicklung implementierungsunabhängig zu modellieren sind. In [LBD02, BDL03] werden die Benutzerrechte beim Design des Klassendiagramms modelliert und in [BKL01, KPP03] werden zusätzlich zum Klassendiagramm eigene Sichten auf die Zugriffsrechte eingeführt. [BP04] sind Vorarbeiten zu dem in dieser Arbeit vorgestellten akteurzentrierten Modell zur Spezifikation von Benutzerrechten. Weitere spezielle Arbeiten zu einer formalen Fundierung von Benutzerrechten beschäftigen sich mit dem Entwicklungsprozess und der Überprüfung von Benutzerrechten in SAP-Anwendungen [HJ03, IBM03].

In [Wim05] wird ein Ansatz zur modellbasierten Entwicklung sicherheitskritischer Systeme vorgestellt, der auf den Ergebnissen der Sicherheitsanalyse sowie der Anforderungsspezifikation aufsetzt und sich mit der Unterstützung der nachfolgenden Phasen (Verifikation anhand von Bedrohungsszenarien, Einführung von Sicherheitsmechanismen, Testfallgenerierung) beschäftigt.

Die Grundprinzipien zu Vorgehensmethoden stammen aus dem Gebiet des Software Engineerings [PB96, Kro97, OHJ+99, Bal96, Bal98, Som00]. Einfluss auf den methodischen Teil der Arbeit hatten nahezu alle modernen objektorientierten Softwareentwicklungstechniken. Im Kontext dieser Arbeit ist die angewendete Methodik MOS [Bre01] hervorzuheben, insbesondere in Zusammenhang mit dem Konzept der Anwendungsfälle. Die hybride Vorgehensweise zur Modellierung von Objekten und Abläufen in Anwendungsfällen basiert auf den Arbeiten von Jacobson [Jac87, JCJO92, JBR99]. Ähnliche Ideen finden sich auch bei [Den91] in Form von Geschäftsvorfällen und mit den Systemoperationen in Fusion [CAB+94].

Die vorgestellte Vorgehensweise basiert auf den neueren generischen, iterativen, objektorientierten Vorgehensmodellen [JBR99, Kru00, DW98, DW99, Boo94]. Der Prozessmuster-basierte Ansatz zur Beschreibung von Vorgehensbausteinen wurde aus Arbeiten von der Process Pattern Community [Amb98, Amb99, GMP+01a, GMP+01b, GMP+02a, GMP+02b, GMP+03a, GMP+03b] übernommen. Die Studien zu den Geschäftsprozessen und zur Unternehmensmodellierung basieren auf den Arbeiten von Hans-Erik Eriksson und Magnus Penker [EP00], Josef Staud [Sta01], Helmut Balzert [Bal98] und Philippe Kruchten [Kru00].

2 Grundlagen

Zu Beginn der Arbeit geben wir einen kurzen Überblick über die beiden Themenschwerpunkte, in die sich diese Arbeit eingliedert. Die Arbeit beschreibt eine Methode zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Systeme.

Eine Methode liefert Vorgaben für ein systematisches Vorgehen bei der Softwareentwicklung. Der Begriff umfasst sowohl einzelne Aktivitäten wie Analyse, Entwurf oder Programmierung als auch die Softwareentwicklung insgesamt [RP97]. Methoden verkörpern eine Sicht der Softwareentwicklung, beziehen sich dabei auf einen Einsatzbereich und geben Richtlinien in Form von Techniken, Mitteln und Organisationsformen an. Auf Vorgehensmodelle, die die Grundlage zu den Methoden bilden und die Prozesse der Softwareentwicklung in überschaubare Abschnitte aufteilen, gehen wir in Abschnitt 2.1 ein.

Das zweite zentrale Themengebiet der Arbeit ist die Zugriffssicherheit. Darunter verstehen wir die Summe aller Maßnahmen, die durchzuführen sind, um unautorisierten Benutzern den Zugriff auf geschützte Systemressourcen zu verwehren. Auf grundlegende Begriffe zur Sicherheit, Ziele der Schutzmaßnamen und Techniken gehen wir in Abschnitt 2.2 genauer ein.

2.1 Vorgehensmodelle

Für komplexe Aufgaben ist es notwendig, den Lösungsprozess in kleinere Einheiten zu gliedern. Ein Vorgehensmodell gibt eine Gliederung für den Prozess der Softwareentwicklung vor, es regelt den Ablauf des Lösungsprozesses und unterteilt ihn in überschaubare Abschnitte, sodass eine schrittweise Planung, Durchführung, Entscheidung und Kontrolle ermöglicht wird (vgl. [PB96]).

Diese Grundidee der Systemtechnik wird auch auf die Vorgangsweise bei der Entwicklung von Software angewandt, Softwareprojekte werden dabei in Phasen unterteilt. Die Menge der Phasen und die Ordnung ihrer zeitlichen Abfolge bezeichnet man als *Softwarelebenszyklus*. In so genannten Phasenmodellen werden die einzelnen Phasen strukturiert hintereinander ausgeführt. In den objektorientierten Lebenszyklusmodellen stehen atomare Modellierungseinheiten mit Verhaltens- und Strukturbeschreibungen im Vordergrund.

2.1.1 Klassisch strukturierte Vorgehensweisen

Das bekannteste und älteste Phasenmodell ist das Wasserfallmodell [Roy70]. In den 70er Jahren entstanden eine Reihe unterschiedlicher Phasenmodelle und kombinierte Phase-/Tätigkeitsmodelle. Den damaligen Modellen lag die Idee zugrunde, dass man ein Problem nur detailliert genug planen müsse, um es bewältigen zu können. Die sequenzielle Abarbeitung der Phasen bewirkt in diesen Modellen, dass Planungs- und Entwicklungsfehler erst spät

bemerkt und Risiken zumeist spät erkannt werden. Veränderungen in der Planung sind kostspielig und schwer zu bewerkstelligen, da die Rückkopplung aus der Implementierungsphase erst sehr spät erfolgt [Rau01].

Ende der 80er Jahre hat Barry Boehm in seinem Spiralmodell [Boe86] die Erkenntnis umgesetzt, dass es einen ganz natürlichen, nicht umgehbaren Wandel bei der Planung und den Anforderungen gibt. Das Spiralmodell ist war immer noch rein sequenziell, jedoch beinhaltet es eine iterative Planung mit integrierter Risikoanalyse. Das Spiralmodell eignet sich nach [PB96] gut für die Einbettung von Qualitätssicherungsmaßnahmen in den Softwareentwicklungsprozess, Fehler können früh erkannt und verschiedene Lösungsmöglichkeiten können mit vertretbarem Aufwand betrachtet werden.

Diese und weitere strukturierte Vorgehensweisen sind ausgehend von der strukturierten Programmierung entstanden und als die erste Generation von Methoden zur Softwareentwicklung von großer Bedeutung (siehe [Bal96]). Im Vordergrund stehen dabei die ablaufbezogene Funktionsmodellierung und die unabhängige Entwicklung eines globalen Datenmodells. In einem eigenen Arbeitsschritt werden Funktions- und Datenmodell aufeinander abgestimmt. Bei diesen Top-Down-Verfahren, d.h., bei Entwicklungen von umfassenden, abstrakten Modellen hin zu programmiersprachlichen Realisierungen, ist der Übergang zwischen Modellebenen oft mit Modellbrüchen verbunden (vgl. [RP97]). Die Top-Down-Vorgehensweise und die Modellbrüche haben in strukturierten Methoden immer wieder zu Problemen geführt, sodass die Entwicklung von neueren Ansätzen vorangetrieben wurde.

2.1.2 Objektorientierte Vorgehensmodelle

Die zu entwickelnden Softwaresysteme wurden über die Jahre immer größer, sodass die Komplexität der Softwareentwicklung stets anstieg und mit der Top-Down-Vorgehensweise der strukturierten Vorgehensweisen nur schwer in den Griff bekommen zu waren. Die Modellbrüche in der funktionsorientierten Softwareentwicklung und der Beginn der objektorientierten Programmierung in den 90er Jahren führten dazu, dass erste Methoden zur objektorientierten Softwarekonstruktion entstanden, wie zum Beispiel [HS93].

Die Objektorientierung stellt dabei die Gegenstände der Anwendungswelt in den Vordergrund, die abstrakt oder konkret sein können. Auf den Gegenständen können Operationen durchgeführt werden, die auf den gekapselten Daten arbeiten. Diese Operationen werden nicht zu standardisierten Abläufen zusammengesetzt, sondern als Dienstleistungen zur Benutzung angeboten. Ein wichtiges Strukturierungsmerkmal für Entwürfe und die Softwarearchitektur sind Konzepthierarchien, die programmiertechnisch durch Vererbung umgesetzt werden (vgl. [RP97]).

Die Objektorientierung bietet die Möglichkeit, Beschränkungen strukturierter Methoden zu überwinden, d.h., Daten zusammen mit Operationen zu modellieren, existierende Systeme flexibel zu erweitern und stückweise zu integrieren. Für alle Ebenen wird dabei eine einheitliche Modellbasis verwendet, Produkte können flexibel strukturiert und in die Umgebung integriert werden.

Die objektorientierte Systemsicht unterstützt insbesondere die Erweiterbarkeit, die Wiederverwendbarkeit und die Wartbarkeit von Systemen (siehe [Bre01]). Die wichtigsten Grundideen objektorientierten Vorgehens sind die Konzepte der Datenkapselung, der Vererbung und

der dynamischen Existenz von Objekten. Objekte sind die atomaren Strukturierungseinheiten eines objektorientierten Systems. Jedes Objekt hat einen internen Zustand, der durch die Ausprägung seiner Attribute bestimmt ist. Dieser interne Datenzustand ist gekapselt, d.h., er ist von außen nicht direkt zugänglich, sondern nur über die Ausführung von Operationen. Sie bilden die Schnittstelle des Objekts. Objekte kommunizieren untereinander durch das Senden von Nachrichten.

Weiterhin sind objektorientierte Systeme hochgradig dynamisch. Objekte können zur Laufzeit erzeugt werden und nach Abschluss der Interaktionen hören sie irgendwann auf, zu existieren. Objekte werden zumeist zu Klassen gruppiert. Klassen verkörpern das Typkonzept und beschreiben Mengen von Operationen mit ähnlicher interner Struktur, Schnittstelle und Verhalten. Eine weitere Gruppierung ähnlicher Objekte wird mit dem Konzept der Vererbung unterstützt.

In den 90er Jahren wurden eine Vielzahl von objektorientierten Softwareentwicklungsmethoden entwickelt. Bekannte, wichtigste Ansätze sind die Methode von Coad und Yourdon [CY91a, CY91b], OMT vom Rumbaugh et al. [RBP+91], die Methoden von Booch [Boo94] und Jacobson [JCJO92] sowie die Methode von Fusion von Coleman et al. [CAB+94].

Eine Vereinheitlichung und Standardisierung der unterschiedlichen Notationen wurde mit der Entwicklung der Unified Modeling Language (UML) [OMG03, BRJ98] unternommen. Methodische Rahmenwerke, die auf Basis der UML definiert wurden, sind beispielsweise der Unified Software Development Process [JBR99], der Catalysis-Ansatz [DW98] oder der Rational Unified Process [Kru00].

2.1.3 Prozessbeschreibung mit Prozessbausteinen

Allumfassende Vorgehensmodelle, wie zum Beispiel das oben vorgestellte Wasserfallmodell [Roy70], sind sehr restriktiv und für Erweiterungen nicht offen, sodass eine methodische Entwicklung von nichtfunktionalen Aspekten, wie beispielsweise die Zugriffssicherheit, nur schwer in den Prozess integriert werden können. In generischen Prozessmodellen, wie zum Beispiel [Kru00, JBR99, DW98] ist der Anpassungsaufwand sehr groß, da die Modelle sehr mächtig und umfangreich sind.

Im Rahmen dieser Arbeit benötigen wir eine Prozessbeschreibung, die es uns ermöglicht, einen kleinen Teil eines Entwicklungsprozesses in den Gesamtprozess einzuordnen und den kleinen Teil detailliert zu beschreiben. Um nicht den Anpassungsaufwand eines generischen Prozessmodells zu unterlaufen, verwenden wir im Rahmen dieser Arbeit Prozessbausteine zur Beschreibung der relevanten Teile eines gesamten Entwicklungsprozesses.

Grundidee des Prozessmuster-basierten Ansatz zur Beschreibung von Vorgängen ist die Beschreibung einzelner Prozessbausteine, in so genannten Mustern [GHJV98, Amb98, Amb99]. Weiterhin sind Prozessmuster ein mächtiges Konzept zur dynamischen Anpassung des Prozessablaufs bei der System- und Softwareentwicklung (siehe hierzu [GMP+01b, GMP+01a, GMP+02a, GMP+03a, GMP+03b]).

Für eine bessere Verständlichkeit und Vergleichbarkeit werden Prozessmuster, wie alle anderen Muster, in einer einheitlichen Form präsentiert [Rau01]. Eine intuitive Beschreibungsform hilft, die Essenz des Musters schnell aufzunehmen. Darüber hinaus liefert eine gute Beschreibung auch alle Fakten, die notwendig sind, um ein Muster anzuwenden sowie die Konsequenzen, die sich aus der Anwendung ergeben. Weiterhin wird durch die Darstellung in Mustern

Tabelle 2.1: Vorlage für die Beschreibung von Prozessmustern

Abschnitt	**Beschreibung des Abschnittsinhalts**
Name	Name des Prozessmusters
Kurzbeschreibung	Kurze Zusammenfassung der Motivation, Ziele und des Hintergrunds des Prozessmusters. Das zentrale Problem wird knapp umrissen.
Problem	Die Entwicklungsaufgabe und/oder das Problem, welches das Prozessmuster adressiert, wird detailliert erklärt und gegebenenfalls werden weitere Einflussfaktoren diskutiert.
Lösung	Der grundlegende Lösungsansatz des Musters wird in diesem Abschnitt dokumentiert. Die Reihenfolge, in der die Aktivitäten auszuführen sind, wird textuell oder grafisch beschrieben.
Aktivitäten	Die Aktivitäten, die im Rahmen der Ausführung des Prozessmusters durchzuführen sind, werden aufgelistet. Der genaue Ablauf der Ausführung der Aktivitäten ergibt sich aus den Beschreibungen zur Lösung und zur Struktur. Dabei können die Aktivitäten auf weitere Prozessmuster verweisen oder eigenständig erklärt sein.
Produktartefakte	In diesem Abschnitt werden die Ein- und Ausgabeprodukte des Prozessbausteins beschrieben. Die Eingabe enthält diejenigen Produktartefakte, die zur Ausführung des Prozessbausteins benötigt werden. Die Liste der Ausgaben enthält die neu erzeugten oder veränderten Produkte.
Struktur	Hier wird eine textbasierte oder grafische Repräsentierung des Lösungswegs des Musters aufgezeigt. Die Abfolge der einzelnen Aktivitäten wird in UML-Aktivitätsdiagrammen oder Übersichtsdiagrammen festgehalten.
Kontext	Bei der Beschreibung des Kontextes geben wir Anforderungen über die Eingabeproduktartefakte hinaus als Voraussetzung für die Ausführung des Musters an. Externe Umstände, Einflüsse oder spezielle Anwendungsrichtlinien werden in diesem Abschnitt betrachtet.
Vor- und Nachteile	Das Prozessmuster selbst und die Konsequenzen für die Anwendung werden kurz diskutiert. Dies erleichtert die Evaluierung der Anwendbarkeit des Musters.
In Beziehung stehende Muster	In einer Liste werden die Prozessmuster, die entweder alternativ, begleitend, im Vorfeld, als nächste Schritte, als Teilmuster oder als übergeordnete Prozessmuster ausgeführt werden können oder mit einer ähnlichen Menge von Produktartefakten arbeiten, kurz beschrieben.
◇	Der Abschluss eines Prozessmusters ist mit einer Raute zur Abgrenzung von nachfolgendem erklärenden Text gekennzeichnet.

die Suche nach möglichen Lösungen erleichtert. In Abhängigkeit vom Problem mit den gegebenen Eingabeartefakten und den gewünschten Ausgabeartefakten kann unter Betrachtung der Vor- und Nachteile das geeignete Muster ausgewählt und die darin vorgeschlagene Lösung abgearbeitet werden.

Alle Muster bestehen stets aus dem Trio von Problem-, Lösungs- und Kontextbeschreibung [GHJV98, Rau01]. Entscheidend dabei ist die Diskussion des Kontexts, der eine detaillierte Problemstellung ermöglicht. Unter Einbeziehung dieses Kontexts kann eine optimale Lösung ausgewählt werden.

Tabelle 2.1 stellt das Beschreibungsformat für Prozessmuster vor, das wir in dieser Arbeit verwenden. Dieses Beschreibungsformat ist angelehnt an die bekannten Musterschablonen aus [Rau01, GMP+01b, GMP+01a, GMP+02a, GMP+03a, GMP+03b].

2.1.4 Die grafische Beschreibungssprache UML

Bei der Unified Modeling Language (UML) [OMG03, BRJ98] handelt es sich um eine Beschreibungssprache zur Spezifikation, Darstellung, Konstruktion und Dokumentation von ganzen Softwaresystemen. Die UML bietet eine Sammlung von Notationen, die sich bei der Modellierung von großen und komplexen Systemen als erfolgreich erwiesen haben.

Die UML ermöglicht nach [BRJ98] eine standardisierte Beschreibung von Systementwürfen auf konzeptionellen und detaillierten Ebenen. Neben der konzeptionellen Beschreibung zum Beispiel von Geschäftsprozessen und Systemfunktionen können auch detaillierte, technische Beschreibungen angefertigt werden. Beispiele hierzu sind Klassenbeschreibungen in einer geeigneten Programmiersprache, Datenbankschemata und wieder verwendbare Softwarekomponenten.

Die UML ist der Nachfolger einer ganzen Anzahl von objektorientierten Analyse- und Designmethoden, die in den späten 80er und frühen 90er Jahren entstanden sind. Insbesondere vereinigt und erweitert sie die Notationen von Grady Booch (siehe [Boo94]), James Rumbaugh (siehe [RBP+91]) und Ivar Jacobson (siehe [JCJO92]). Nach der ersten Phase der Diversifikation folgte eine Phase der Vereinheitlichung der objektorientierten Methoden, der auch die OMG (Object Management Group) forciert wurde. Die UML 1.5 ist seit März 2003 verfügbar und stellt derzeit die aktuelle Version dar.

2.2 Grundlagen zur IT-Sicherheit

Sicherheit ist ein sehr umfassendes Thema, das auch außerhalb der IT eine wichtige Rolle spielt, so denke man beispielsweise an den Schutz von Kernkraftwerken oder Flugzeugen vor terroristischen Angriffen sowie an die innere Sicherheit eines Staates.

Die IT-Sicherheit als Teilgebiet der allgemeinen Sicherheit ist ebenfalls ein sehr weit reichendes Gebiet, das sich beispielsweise mit

- *technischen Lösungsmöglichkeiten*, wie Protokollen, Authentifikationstechniken, Schlüsselmanagement, Hashfunktionen oder Realisierungen der Zugriffskontrolle, etc.,

- *anwendungsspezifischen Fragestellungen*, wie zum Beispiel Netzwerksicherheit und Datenbanksicherheit sowie
- mit *übergeordneten Sicherheitsthemen*, wie beispielsweise Überwachung, Passwörter, Biometrie, Softwareanomalien, Konstruktion sicherer Systeme, Datenschutz und Kopierschutz oder Steganaografie

beschäftigt (siehe [And01, Bau97, Eck03, Sch00, VM02]). Im Folgenden stellen wir eine kleine Auswahl an grundlegenden Begriffen, Schutzzielen, Sicherheitsprinzipien, Bedrohungen und Angriffe sowie Sicherheitsgrundfunktionen vor. Viele der Begriffe und Definitionen wurden aus dem sehr umfangreichen Buch [Eck03] zur IT-Sicherheit von Frau Prof. Dr. C. Eckert übernommen und, wenn notwendig, angepasst.

2.2.1 Grundlegende Begriffe

Bevor wir auf Ziele, Prinzipien und Funktionen der Sicherheit eingehen, sind grundlegende Begriffe und der Sicherheitsbegriff zu definieren.

IT-System

Ein IT-System ist ein geschlossenes oder offenes, dynamisches technisches System mit der Fähigkeit zur Speicherung, Verarbeitung, Übertragung und Darstellung von Informationen.

Information und Datenobjekte

IT-Systeme speichern, verarbeiten, übertragen und stellen Informationen dar. Die Information ist ein Abstraktum, das in Form von Daten bzw. Datenobjekten repräsentiert wird. Die Information, die durch ein Datum repräsentiert wird, ergibt sich aus einer festgelegten Interpretationsvorschrift.

Sicherheit

Die *Funktionssicherheit* (engl. *safety*) ist der Zustand eines Systems, frei von unvertretbaren, schadenverursachenden Risiken zu sein. Das System ist frei von Gefahren, die vom Betrieb der Hardware oder Software ausgehen und die der Umwelt drohen (DIN EN 61508-4).

Anders ausgedrückt ist die Funktionssicherheit die Sicherheit gegen Fehler und unbeabsichtigte Unfälle. Ein Unfall ist dabei ein unerwünschtes und ungeplantes, nicht unbedingt unerwartetes Ereignis, welches einen Grad von Verlusten zur Folge hat. Ein Verlust ist eine Zerstörung oder Schädigung von Einrichtungen bzw. Systemen oder Verletzung bzw. Tod von Lebewesen, insbesondere von Menschen. Ein Fehler in einem System ist ein Zustand, der durch Anregungen aus der Umwelt des Systems zu einem Unfall oder Verlust führen kann.

Die *Informationssicherheit* (engl. *security*) ist die Eigenschaft eines funktionssicheren Systems, nur solche Systemzustände anzunehmen, die zu keiner unautorisierten Informationsveränderung oder -gewinnung führen.

Die *Zugriffssicherheit* ist die Eigenschaft eines funktionssicheren Systems, nur solche Systemzustände anzunehmen, die zu keinem unautorisierten Zugriff auf Systemressourcen und Daten führen. Hierbei müssen die Schutzziele Vertraulichkeit, Integrität, Verbindlichkeit und

Verfügbarkeit angestrebt werden (ISO/EEC 17799). Im Gegensatz zur Datensicherheit (engl. *protection*) bleiben bei der Zugriffssicherheit Aspekte des Datenschutzes und der Datensicherung außen vor.

Vereinfacht ausgedrückt ist die Zugriffssicherheit die Sicherheit gegen absichtliche Angriffe.

2.2.2 Schutzziele

In zugriffssicheren Systemen können die Angriffsziele in Kategorien aufgeteilt werden. In der Literatur haben sich hierzu die Begriffe Schutzziele [Eck03], Sicherheitsdienste (engl. *security services*) [MFW+99] und Sicherheitseigenschaften etabliert. Weiterhin werden sie auch als Dimensionen der Sicherheit (engl. *dimensions of security*) [Dos01], als Aspekte [BSW01] oder als kryptografische Ziele (engl. *cryptographic goals*) [MvOV96] bezeichnet.

Authentizität

Unter der Authentizität eines Objektes bzw. Subjekts (engl. *authenticity*) verstehen wir die Echtheit eines Objekts bzw. Subjekts, die anhand seiner eindeutigen Identität und seiner charakterisierenden Eigenschaften überprüfbar sind.

Unter Identität einer Person oder Sache wird die Summe der Merkmale verstanden, anhand derer sich diese von anderen unterscheidt.

Authentifikation

Die Authentizität eines Subjekts bzw. Objekts wird durch Maßnahmen zur Authentifikation (engl. *authentication*) überprüft. Dazu muss nachgewiesen werden, dass eine behauptete Identität eines Objekts oder Subjekts mit dessen charakteristischen Eigenschaften übereinstimmt.

Eine Authentifikation wird in herkömmlichen Systemen meist nur für Benutzer als Subjekte durchgeführt. Die Identifikation basiert auf der Vergabe von eindeutigen Benutzerkennungen (engl. *account*) oder Benutzernamen. Charakteristische Eigenschaften zum Nachweis der Identität sind beispielsweise Passworte, Chipkarten, biometrische Merkmale, etc.

Autorisierung

Autorisierung ist die Prüfung der Berechtigung eines Subjekts zum Zugriff auf eine Information beziehungsweise auf ein Objekt.

Integrität

In einem IT-System ist die Integrität (engl. *integrity*) gewährleistet, wenn es Subjekten nicht möglich ist, die zu schützenden Daten unautorisiert und unbemerkt zu manipulieren.

Die Integrität erfordert im System eine Rechtefestlegung zur Nutzung von Daten. Für die Erkennung unautorisierter Manipulationen müssen geeignete Techniken eingesetzt werden.

Der Begriff der Integrität ist nicht zu verwechseln mit dem Begriff der Datenintegrität von Datenbanksystemen. Im Umfeld von Datenbanksystemen versteht man unter dem Begriff der Datenintegrität oder kurz Integrität die Gewährleistung der Datenkonsistenz im DB-System, insbesondere die eindeutige Zuordnung von Primär- und Fremdschlüsseln (referentielle Integrität) [KE01].

Vertraulichkeit

In einem System ist die Informationsvertraulichkeit, oder kurz die Vertraulichkeit, (engl. *confidentiality*) gewährleistet, wenn auf Informationen nur von berechtigten Subjekten oder Objekten zugegriffen werden kann.

Verbindlichkeit

Die Verbindlichkeit oder Zuordenbarkeit (engl. *non repudiation*) einer Menge von Aktionen ist gewährleistet, wenn es nicht möglich ist, dass ein Subjekt im Nachhinein die Durchführung einer solchen Aktion abstreiten kann.

Verfügbarkeit

Die Verfügbarkeit (engl. *availability*) eines Systems sagt aus, dass authentifizierte Subjekte in der Wahrnehmung ihrer Berechtigungen nicht unautorisiert beeinträchtigt werden können.

2.2.3 Sicherheitsprinzipien

Bereits 1975 haben Saltzer und Schroeder [SS75] allgemeine Prinzipien zur Entwicklung von sicherer Software entwickelt. Weitere, allgemeinere Prinzipien zur Entwicklung sicherer Systeme sind auch in [VM02] zu finden.

Erlaubnisprinzip

Nach dem Erlaubnisprinzip (engl. *fail-safe defaults principle*) ist grundsätzlich jeder Zugriff verboten und nur durch eine explizite Erlaubnis kann ein Zugriffsrecht gewährt werden.

Vollständigkeitsprinzip

Das Vollständigkeitsprinzip (engl. *complete mediation principle*) fordert die Überprüfung jedes Zugriffs auf seine Zulässigkeit. Ein System, in dem Zugriffsrechte zur Nutzung von Dateien vergeben werden und eine Rechteüberprüfung nur beim Öffnen einer Datei durchgeführt wird, erfüllt dieses Vollständigkeitsprinzip nicht. Es wäre hier möglich, dass ein Benutzer eine Datei über eine längere Zeit geöffnet hält und auf die Datei zugreifen kann, obwohl er zwischendurch die Rechte darauf verloren hat.

Prinzip der minimalen Rechte

Durch das Prinzip der minimalen Rechte (engl. *need-to-know principle*) enthält ein Subjekt nur genau die Zugriffsrechte, die es zur Erfüllung seiner Aufgaben benötigt. Ein Superuser mit uneingeschränkten Rechten im System ist ein offensichtliches Gegenbeispiel zu diesem Prinzip.

Prinzip der Benutzerakzeptanz

Dass die eingesetzten Sicherheitsmechanismen einfach zu nutzen sind und dass sie automatisch und routinemäßig angewendet werden, wird durch das Prinzip der Benutzerakzeptanz (engl. *economy of mechanism prinicple*) gefordert.

Offener Entwurf

Das Prinzip des offenen Entwurfs (engl. *open design*) fordert, dass die im Entwurf eingesetzten Mechanismen und Verfahren offen gelegt werden müssen (*no security through obscurity*). Die Sicherheit eines Systems darf nicht von der Geheimhaltung spezieller Verfahren abhängig sein, zum Beispiel darf die Sicherheit kryptografischer Verfahren nicht darauf beruhen, dass die Verschlüsselungsverfahren nicht bekannt sind.

Sicherheitskern

Bei der Konstruktion von großen Systemen, wie beispielsweise bei Web-basierten Informationssystemen und Betriebssystemen, werden sicherheitsrelevante Dienste und Maßnahmen häufig von den übrigen Systemteilen isoliert in einem Sicherheitskern (engl. *security kernel*) realisiert. Die vertrauenswürdigen Sicherheitsdienste nennt man die *Trusted Computer Base* (TCB). Eine TCB ist definiert als eine Menge von Komponenten (Hardware, Software, Personen, etc.), deren korrekte Funktionsweise ausreichend ist, um die Sicherheitspolitik des Systems zu erreichen. Ein Fehlschlagen der TCB würde einen Bruch der Sicherheitspolitik bedeuten.

2.2.4 Bedrohungen und Angriffe

Ein IT-System kann verschiedene Schwachstellen besitzen, die zur Beeinträchtigung der Datenintegrität, der Informationsvertraulichkeit und der Verfügbarkeit ausgenutzt werden können. Im Folgenden führen wir die Begriffe Bedrohung und Angriff ein und gehen auf spezielle Angriffe durch Viren, Würmer, trojanische Pferde und mobilem Code näher ein.

Bedrohung

Schwachstellen (engl. *weakness*) eines Systems sind diejenigen Punkte, an denen ein System verwundbar ist. Eine Verwundbarkeit (engl. *vulnerability*) ist eine Schwachstelle, über welche die Sicherheitsdienste des Systems umgangen, getäuscht oder unautorisiert modifiziert werden können.

Eine Bedrohung (engl. *threat*) des Systems zielt darauf ab, ein oder mehrere Schwachstellen oder Verwundbarkeiten auszunutzen, um einen Verlust der Datenintegrität, der Informationsvertraulichkeit oder der Verfügbarkeit zu erreichen oder um die Authentizität von Subjekten zu gefährden.

Das Risiko (engl. *risk*) einer Bedrohung ist die Wahrscheinlichkeit des Eintritts eines Schadensereignisses und die Höhe des potentiellen Schadens, der dadurch hervorgerufen werden kann.

Angriff

Ein Angriff (engl. *attack*) ist ein nicht autorisierter Zugriff bzw. Zugriffsversuch auf das System. Dabei werden aktive und passive Angriffe unterschieden. Passive Angriffe betreffen die unautorisierte Informationsgewinnung und zielen auf den Verlust der Vertraulichkeit ab, wie beispielsweise das Abhören von Datenleitungen in vernetzten Systemen. Bei aktiven Angriffen werden unautorisierte Modifikationen an Datenobjekten ausgeführt, wie zum Beispiel das Entfernen von Paketen aus einem Datenstrom. Sie richten sich gegen die Datenintegrität oder Verfügbarkeit eines IT-Systems.

Passiven Angriffen kann durch die Anwendung von kryptografischen Verfahren entgegengewirkt werden, wie beispielsweisedurch die verschlüsselte Übertragung von Passwörtern.

Aktive Angriffe können durch eine Beschränkung der Benutzerrechte zur Durchführung von Schreibzugriffen verhindert werden. Hierbei hilft der Einsatz von Sicherheitsmodellen, die es erlauben, von Realisierungsdetails zu abstrahieren und sich auf das Wesentliche konzentrieren, hier beispielsweiseauf die Vergabe von Schreibrechten.

Angriffe durch Viren, Würmer, trojanische Pferde und mobilem Code

Angriffe durch Computerviren, Würmer, trojanische Pferde und mobilem Code gehören zu den Bedrohungen, die in heutigen Systemen am häufigsten auftreten. Es handelt sich dabei um Programme, die im Hintergrund bestimmte Funktionen ausführen. Diese Programme werden auch als Schadsoftware (engl. *malware*) bezeichnet. Die beabsichtigte, böswillige Funktionalität in Schadsoftware unterscheidet sich vom Fehlverhalten aus unbeabsichtigten Programmierfehlern (engl. *bugs*).

Ein *Computervirus* ist eine Befehlsfolge, die zur Ausführung ein Wirtsprogramm benötigt. Bei der Ausführung eines Virus wird eine Kopie (Reproduktion) oder eine modifizierte Version des Virus in einen Speicherbereich geschrieben (Infektion). Zusätzlich zur Fähigkeit der Reproduktion enthalten Viren in der Regel einen Schadanteil, der bedingt oder unbedingt durch einen Auslöser aktiviert werden kann. Viren können in Programm-, Boot-, Makro- und Datenviren unterteilt werden. Maßnahmen gegen den Angriff von Computerviren sind beispielsweisedie Anwendung von Virenscanner, die Beschränkung der Benutzerrechte eines Rechners, das Verschlüsseln von Programmen und Daten sowie der Einsatz von Hashfunktionen zur Überprüfung von Manipulationen.

Ein *Wurm* ist ein ablauffähiges Programm mit der Fähigkeit der Reproduktion. Ein Programm eines Wurms besteht in der Regel aus mehreren Programmteilen, den Wurmsegmenten. Die Vervielfältigung erfolgt selbstständig unter Kommunikation mit anderen Wurmsegmenten. Würmer bedrohen die Integrität und Vertraulichkeit sowie die Verfügbarkeit von Rechnern. Würmer beanspruchen zumeist viele Ressourcen, insbesondere eine hohe Netzlast. Würmer nutzen zur Verbreitung Lücken im System und in der Anwendungssoftware. Das Eindringen kann durch eine geeignete Konfiguration, beispielsweisedurch Beschränkung des Diensteangebots, verringert oder ausgeschlossen werden. Durch eine restriktive Vergabe von Zugriffsberechtigungen, insbesondere auf Passwörter, lässt sich unerlaubter Informationsgewinn und das Eindringen von fremdem Code beschränken.

Als *Trojanisches Pferd* bezeichnen wir ein Programm, dessen implementierte Ist-Funktionalität nicht mit der angegebenen Soll-Funktionalität übereinstimmt. Im Gegensatz zu Systemen, bei denen die *Safety* nicht gewährt wird, erfüllen diese Systeme die Soll-Funktionalität, besitzen jedoch eine darüber hinausgehende, beabsichtigte zusätzliche aber verborgene Funktionalität. So genannten Trojanern kann ebenfalls durch die Beschränkung der Benutzerrechte auf sensible Daten, wie Passworte, PINs und TANs sowie Programmen entgegengewirkt werden (Prinzip der minimalen Rechte). Werden Programme vom Hersteller signiert, so können nachträgliche Manipulationen erkannt werden.

Unter dem Begriff des *mobilen Codes* lassen sich alle Programme zusammenfassen, deren Code auf entfernten, partiell nicht vertrauenswürdigen Rechnern generiert wurden und die auf Gastrechnern (engl. *hosts*) ausgeführt werden. Der mobile Code, der bei der Übertragung

angegriffen werden kann, ist durch Verschlüsselung zu schützen. Der Gastrechner, auf dem der mobile Code ausgeführt wird, kann durch den Einsatz eines isolierten Speicherbereichs (Sandbox), innerhalb dessen der mobile Code freien Zugriff besitzt, geschützt werden.

2.2.5 Sicherheitsgrundfunktionen

Mit den Sicherheitsgrundfunktionen aus [Eck03] steht ein Baukasten von allgemeinen Maßnahmen zur Abwehr von Bedrohungen zur Verfügung. Diese Grundfunktionen sind entsprechend der individuellen Anforderungen eines zu konstruierenden Systems zu kombinieren und realisieren. Die für diese Arbeit relevanten Sicherheitsgrundfunktionen listen wir im Folgenden auf.

Identifikation und Authentifikation

Um Bedrohungen, die sich aus Maskierungsangriffen ergeben, abwehren zu können und um generell unautorisierte Zugriffe zu unterbinden, müssen Subjekte und sicherheitsrelevante Objekte im System eindeutig identifizierbar sein. Die Identität ist bei der Authentifikation anhand der charakteristischen Eigenschaften des Subjekts bzw. Objekts zu überprüfen. Für die Authentifikation ist festzulegen, wann diese durchzuführen ist. Beispielsweise findet bei Betriebssystemen eine Überprüfung nur beim Systemzugang statt. Für sicherheitsrelevante Aktionen kann es aber nötig sein, dass die Identität von Subjekten erneut zu überprüfen ist.

Rechteverwaltung

Die Rechteverwaltung liefert die Basis zur Abwehr von Bedrohungen der Integrität und Vertraulichkeit von Objekten infolge unautorisierter Zugriffe. Subjekte besitzen Rechte zum Zugriff auf Objekte. Aus den Sicherheitsanforderungen ist zu ermitteln, welche Rechte für zu schützende Objekte festzulegen sind. In der Rechteverwaltung ist darüber hinaus festzulegen, unter welchen Umständen ein Subjekt ein vergebenes Recht wahrnehmen darf, beispielsweise Mitgliedschaft in einer Rolle.

Rechteprüfung

Die Zugriffskontrolle ist ein weiteres notwendiges Instrument zur Abwehr von unautorisierten Zugriffen. Mit der Sicherheitsgrundfunktion der Rechteprüfung ist festzulegen, wann, d.h., bei welchen Aktionen, eine Überprüfung stattfinden muss. Gemäß dem Vollständigkeitsprinzip sollte jeder Zugriff kontrolliert werden. Aus Leistungs- und Aufwandsgründen wird davon aber häufig abgewichen.

Beweissicherung

Die Beweissicherung ist zur Abwehr von Angriffen notwendig, die versuchen, durchgeführte Aktionen im Nachhinein abzustreiten. Dabei ist festzulegen, welche Ereignisse zu protokollieren und welche Informationen dabei zu erfassen sind. Minimal müssen dabei die Identität des ausführenden Subjekts, die betroffenen Objekte und Operationen sowie der Zeitpunkt der Ausführung protokolliert werden. Mit der Beweissicherung ist darüber hinaus festzulegen, welche Subjekte unter welchen Randbedingungen auf die protokollierten Daten zugreifen dürfen. Zugriffe auf protokollierte Daten sind restriktiv zu gewähren, um eine nachträgliche unautorisierte Modifikation zu verhindern.

3 Eine Fallstudie: Das Projektverwaltungssystem TimeTool

Im vorausgehenden Kapitel haben wir einen Überblick über die Themengebiete Vorgehensmodelle und Sicherheit gegeben. Da existierende Vorgehensmodelle kaum auf die Entwicklung von Aspekten der Zugriffssicherheit in den frühen Phasen eingehen, müssen für die Einführung eines Vorgehensmodells in den Phasen der Anforderungsspezifikation Prozessabläufe definiert, Beschreibungstechniken und Produktartefakte erweitert und angepasst sowie ein Modell zur frühen Spezifikation von Benutzerrechten entwickelt werden.

Die eigens definierten Prozessabläufe beschreiben dabei das Vorgehen in den Phasen der Anforderungsspezifikation. Prozesse benötigen zur Prozessabarbeitung Produktartefakte und erweitern bestehende oder erstellen neue Produkte. Im Rahmen unseres Vorgehensmodells sind die Produkte zur Beschreibung von Sicherheitsaspekten, die während der Phasen der Anforderungsspezifikation erstellt und erweitert werden, von primärem Interesse. Intuitive, erweiterte Beschreibungstechniken sind notwendig, um Sicherheitsaspekte in den verschiedenen frühen Phasen angemessen darstellen zu können, da derzeitige Beschreibungstechniken, wie beispielsweise [FS98, OMG03, BRJ98, Jür04], dies kaum unterstützen. Ein Modell zur Spezifikation von Benutzerrechten benötigen wir für eine frühzeitige Formalisierung und damit einer kontextbezogenen Klärung informaler Aussagen über Benutzerberechtigungen.

Ein Benutzerrechtemodell, Prozessabläufe mit ihren Produktartefakten sowie erweiterte Beschreibungstechniken sind Gegenstand der folgenden Kapitel. Die Themen in diesen Kapiteln sind teilweise von theoretischer und abstrakter Natur. Da Prozessmodelle in der Praxis jedoch nur dann eingesetzt werden, wenn sie verständlich und nachvollziehbar sind und somit den Bezug zur Praxis nicht verlieren, werden wir die theoretischen Inhalte anhand eines konkreten Anwendungsbeispiels illustrieren.

In diesem Kapitel stellen wir die Fallstudie, das Projektverwaltungssystem *TimeTool*, vor. *TimeTool* unterstützt Verwaltungsaufgaben eines Projekts, insbesondere zur Abrechnung und Kontrolle von Bearbeitungszeiten.

Zunächst erläutern wir in Abschnitt 3.1 die Motivation und den Hintergrund der Fallstudie *TimeTool*. Eine kurze Systembeschreibung sowie eine Darstellung der Anwendungsfälle des *TimeTool*-Systems wird im Abschnitt 3.2 gegeben. In der Zusammenfassung in Abschnitt 3.3 motivieren wir den Einsatz der Fallstudie als Anwendungsbeispiel in dieser Arbeit.

3.1 Motivation und Hintergrund

Die Fallstudie des Projektverwaltungssystems *TimeTool* stammt ursprünglich aus den Übungen zur Vorlesung "Softwareentwicklung 4 – Projektorganisation", die im Sommersemester 2003 an der Universität Innsbruck abgehalten wurden. Dabei galt es, ein kleines Projektmanagementwerkzeug zu erstellen.

Ziel des Projektes war die Realisierung eines Projektverwaltungssystems in einem Team, in einer praxisorientierten Entwicklungsumgebung und unter Verwendung moderner Entwurfsmethoden (vgl. [BBEH03]).

Da es sich bei der Entwicklung um ein relativ kleines System handelt, basiert die Entwicklung auf dem sequenziellen Wasserfallmodell. In der *Anforderungsspezifikation* wurden die fachlichen Anforderungen an das System und der Systemumfang identifiziert (vergleiche auch Kapitel 3.2), ein Oberflächenprototyp erstellt und ein Projektplan mit Terminplanung und Aufwandsschätzung angefertigt. Im *Testdrehbuch* wurden die Testfälle für den Abnahmetest beschrieben. Im *Softwaredesign* entstanden die grobe Komponentenstruktur der Implementierung als Softwarearchitektur sowie ein Projektplan mit Arbeitspaketen. Nach der *Implementierung* vervollständigten die *Dokumentation* und ein *Abnahmetest* die Entwicklung.

Abbildung 3.1 zeigt Screenshots zum Projektverwaltungssystem *TimeTool*, insbesondere die Dialoge *Login*, *Projektauswahl*, *Projektbuchung* und *Mitarbeiterstammdaten*.

Da es sich bei dem Projektverwaltungssystem *TimeTool* um ein kleines und überschaubares System handelt und es sich für die Darstellung sowohl vorgehensspezifischer Aspekte eignet als auch die Integration von Sicherheitseigenschaften erlaubt, haben wir uns entschlossen, dieses System als Fallstudie in den weiteren Kapiteln dieser Arbeit zu verwenden.

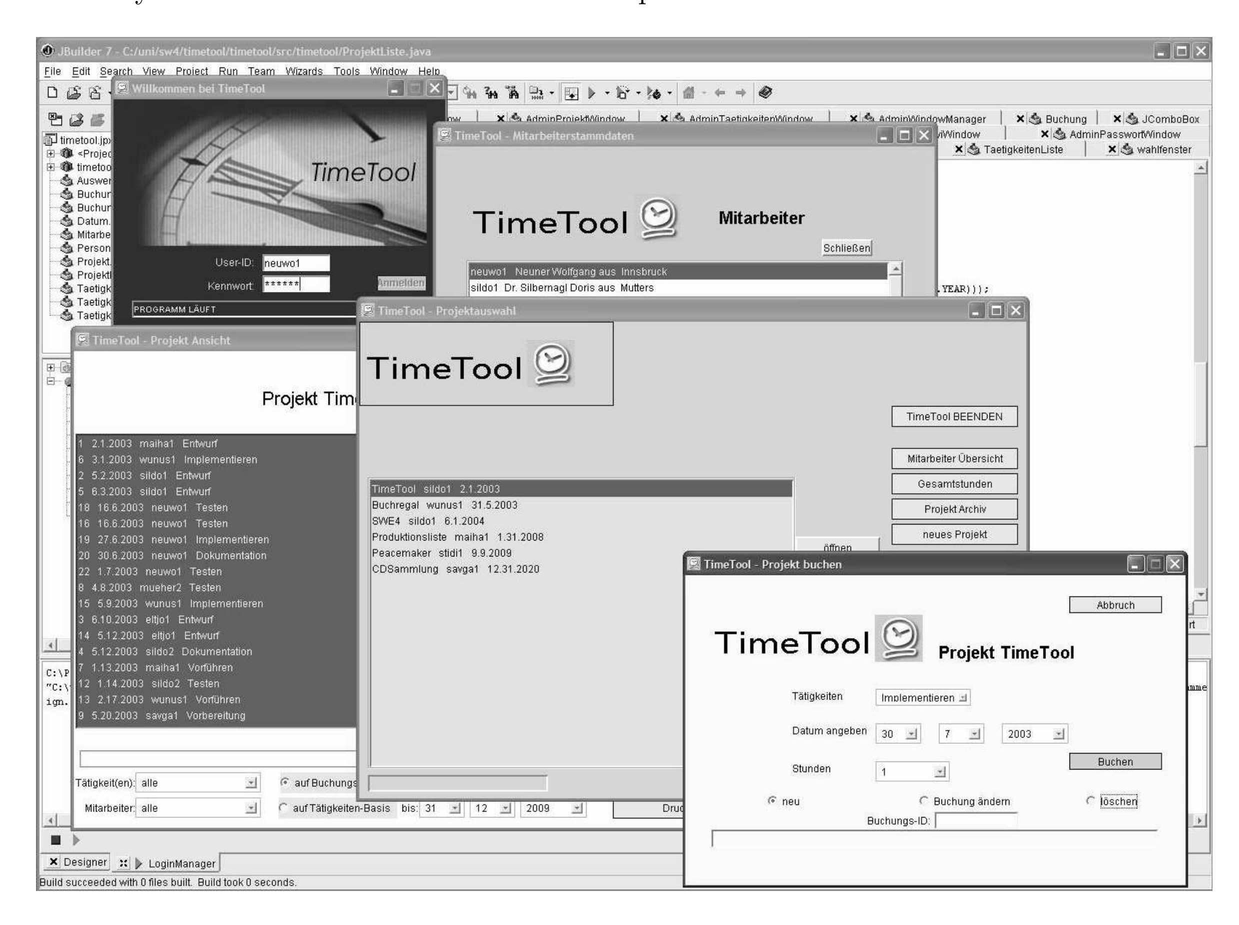


Abbildung 3.1: Das Projektverwaltungssystem TimeTool

3.2 Systembeschreibung und Anwendungsfälle

Projektmanagementwerkzeuge unterstützen das Wissen und die Erfahrung bei der Planung und Steuerung von Projekten und steigern somit die Effizienz der Projektarbeit. Sie unterstützen eine Projektabwicklung in Bezug auf Termin-, Ressourcen-, Kosten-, Konfigurations-, Risiko-, Dokumentations- und Informationsmanagement.

Aus dieser Vielzahl von Leistungen, die in einem Projektmanagementwerkzeug integriert werden können, umfasst unser Projektverwaltungssystem *TimeTool* nur einen kleinen Ausschnitt, der innerhalb der Übungen während eines Semesters bearbeitet werden konnte. Aus diesem Grund befasst sich *TimeTool* nur mit Ausschnitten aus dem Termin- und Ressourcenmanagement.

In der Praxis wird eine Projektdurchführung auf kleine Teilaufgaben abgebildet, so genannte Aktivitäten. Diese Aktivitäten werden von einem oder mehreren Mitarbeitern bearbeitet. Die Aufgabe eines Projektmanagementwerkzeug besteht unter anderem darin, dass die Mitarbeiter zu den bearbeiteten Teilaufgaben ihre benötigten Stunden in so genannten Stundenbuchungen eintragen können. Somit ist es möglich, den aktuellen Projektfortschritt zu beobachten und eventuelle Zeitverzögerungen und Budgetüberschreitungen frühzeitig zu erkennen.

Das Projektverwaltungssystem *TimeTool* ermöglicht es beispielsweise, den Mitarbeitern in einem Projekt die geleisteten Stunden zu einer Produktaktivität zu buchen, so kann zum Beispiel *der Mitarbeiter K. Burger 8 Stunden für die Aktivität Testen im Projekt OnlineBanking* buchen. *TimeTool* ist nicht nur auf ein Projekt beschränkt, es unterstützt somit das Einrichten von neuen Projekten mit zugeordneten Projektaktivitäten. Da ein Projektablauf bereits zu Beginn eines Projektes geplant wird und somit die Einzelaktivitäten in ihrer Dauer bereits festgelegt sind, wird bei der Definition der Projektaktivitäten der geplante Aufwand sowie ein geplanter Start- und Endtermin festgelegt.

Das Projektverwaltungssystem erlaubt zudem mehreren Benutzergruppen, so genannten Rollen, den Zugang zum System. Neben *Projektmitarbeitern* sind spezielle Zugänge für die Rollen *Projektleiter* und *Administrator* vorhanden, weitere Benutzer werden in der Rolle *Andere* zusammengefasst. *TimeTool* unterschützt auch verschiedene Sichten auf die Projektdaten. Zum Beispiel kann sich ein *Projektmitarbeiter* die geleisteten Stunden zu einem Projekt oder ein *Projektmanager* die Gesamtheit der bereits verbuchten Stunden anzeigen lassen.

Das mehrbenutzerfähige System *TimeTool* wurde in Java mit einer Swing-Oberfläche und einer Oracle-Datenbank realisiert.

3.2.1 Systemüberblick

Wie bereits beschrieben, befassen sich die umgesetzten Funktionen in erster Linie mit der Termin- und Ressourcenplanung. Zielgruppe des Systems *TimeTool* sind die Akteure Projektmitarbeiter (*Team Worker*), Projektmanager (*Project Manager*), *Administrator* und Andere (*Others*).

Die derzeitige Realisierung des Systems umfasst im Wesentlichen die Funktionen *Anmeldung im System*, *Verwaltung von Benutzerdaten*, *Verwaltung von Projektdaten*, *Planen von Projekten*, *Analysieren von Projekten*, *Buchen von Stunden für Projektaktivitäten*, *Anzeige von Projektdaten* sowie die *Archivierung von Projekten.*

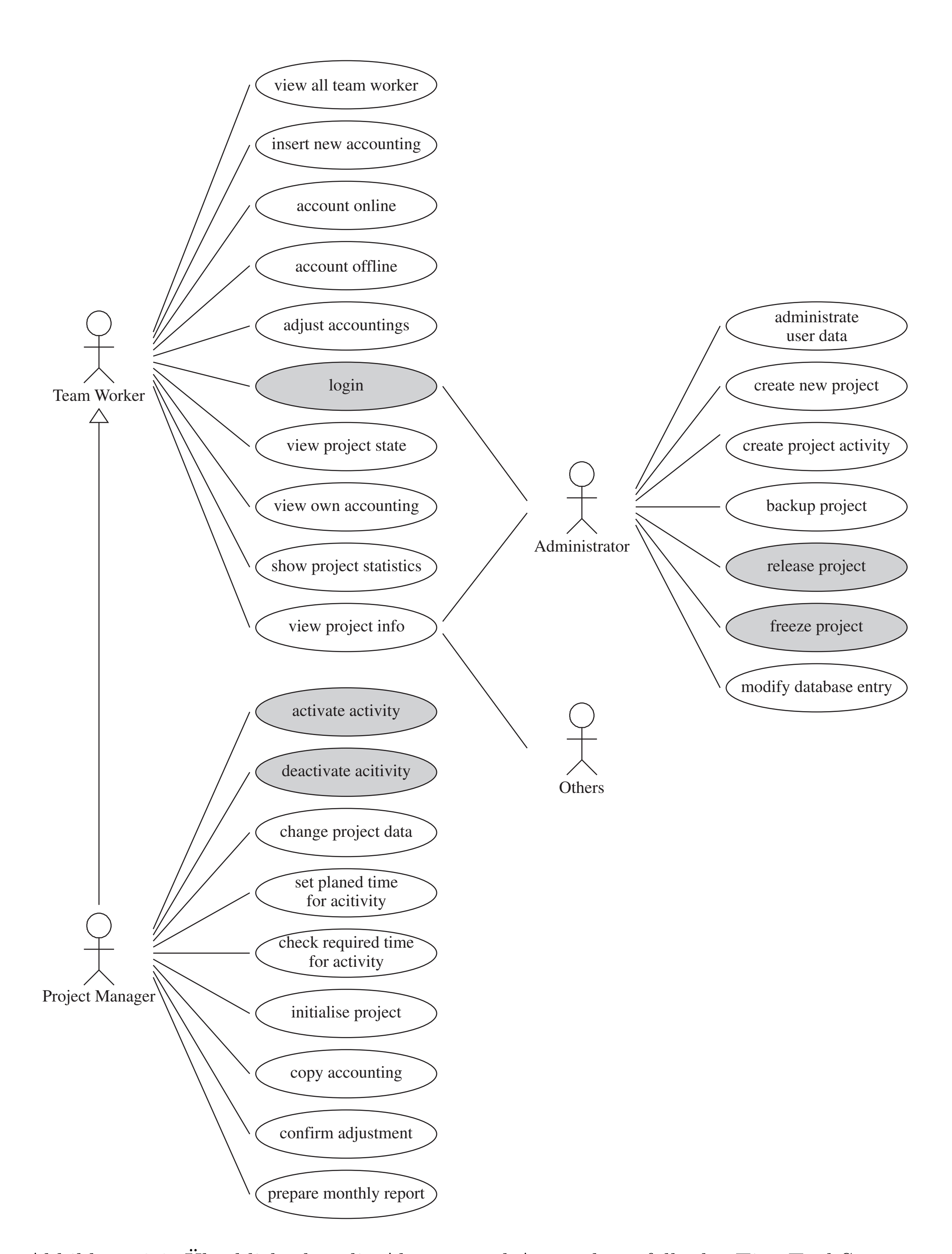


Abbildung 3.2: Überblick über die Akteure und Anwendungsfälle des TimeTool-Systems

Neben diesen fachlichen Anforderungen wird eine *gute Benutzbarkeit*, *hohe Zuverlässigkeit* und *hohe Sicherheit* dem System abverlangt.

3.2.2 Anwendungsfälle

Die aus den, im Systemüberblick dargestellten, Produktfunktionen abgeleiteten Anwendungsfälle sind in Abbildung 3.2 in einem UML-Anwendungsfalldiagramm (engl. *Use Case Diagram*) [OMG03, FS98, BRJ98] dargestellt. Dabei wurde eine Zuordnung der Akteure, welche als Zielgruppe festgelegt wurden, zu den Produktfunktionen geschaffen. Die Produktanforderungen wurden gruppiert und einer konkreten Rolle zugeordnet.

Eine Besonderheit stellt der Akteur *Project Manager* dar. Da er sowohl in der Rolle des *Team Workers* als auch in der Rolle des *Project Managers* arbeiten kann, kann er sowohl seine rollenspezifischen Anwendungsfälle als auch die der Rolle *Team Worker* ausführen. Im *Use Case Diagram* [BRJ98] ist diese Besonderheit dadurch gekennzeichnet, dass der *Project Manager* mit einer Vererbungs-Assoziation mit dem *Team Worker* verbunden ist.

Bei der Beschreibung der Anforderungen wurden hier nur sehr offensichtliche Anwendungsfälle zur Gewährleistung der Zugriffssicherheit mit aufgenommen, wie beispielsweise der Anwendungsfall *login* oder die Anwendungsfälle *freeze* und *release project* sowie *activate* und *deactivate activity*. Diese Sicherheitsanwendungsfälle sind in Abbildung 3.2 grau hinterlegt. Weitere Anwendungsfälle zur Gewährleistung der Zugriffssicherheit sind bei der Analyse der Anforderungen aus dem globalen Sicherheitsziel “hohe Sicherheit” zu ermitteln. In Hinblick auf eine akteurzentrierte Spezifikation der Benutzerrechte wurden die Anwendungsfälle den ausführenden Rollen zugeteilt.

3.3 Zusammenfassung

Für ein Vorgehensmodell für zugriffssichere Systeme sind spezielle Prozessabläufe, erweiterte Beschreibungstechniken, Produkte zur Beschreibung der Sicherheitseigenschaften und ein Modell zur Spezifikation von Benutzerrechten notwendig. Für eine anschauliche Erarbeitung und Einführung dieser Themen und deren Zusammenhänge haben wir das Anwendungsbeispiel des Projektverwaltungssystems *TimeTool* eingeführt.

TimeTool realisiert einen Ausschnitt eines Projektmanagementtools, insbesondere Teilaufgaben zum Termin- und Ressourcenmanagement. Das System unterscheidet zwischen verschiedenen Akteuren und stellt in Abhängigkeit von der Benutzerrolle verschiedene Systemfunktionalitäten zur Verfügung.

Obwohl das Projektverwaltungssystem *TimeTool* nur sehr begrenzte Funktionen eines Projektmanagementtools bereitstellt, eignet es sich sehr gut für unsere Betrachtungen im Rahmen dieser Arbeit. Wie wir noch sehen werden, ist durch die eingeschränkte Funktionalität das System übersichtlich, sodass beispielsweise die Systemfunktionen und das Klassendiagramm überschaubar bleiben. Weiterhin bietet dieses kleine System bereits eine große Menge an Sicherheitsanforderungen, die wir in den folgenden Kapiteln systematisch herausarbeiten und in den Phasen der Anforderungsspezifikation integrieren werden.

4 Akteurzentriertes Modell zur Spezifikation von Benutzerrechten

Nachdem wir in den vorausgehenden Kapiteln die Grundlagen zu Vorgehensmodellen und zur Sicherheit sowie die Fallstudie präsentiert haben, stellen wir hier das akteurzentrierte Modell zur Spezifikation von Benutzerrechten auf Basis eines objektorientierten Systemmodells vor.

Die Spezifikation der Benutzerrechte findet heute in den gängigen Methoden am Ende der Designphase statt, d.h., nach Fertigstellung des Klassendiagramms. Die Benutzerrechte werden dabei meist nur informal spezifiziert. Da aber alle Sicherheitseigenschaften während der gesamten Entwicklung durchgängig, sukzessive und gemeinsam mit der Funktionalität zu entwickeln sind, muss auch mit der Modellierung der Benutzerrechte bereits frühzeitig begonnen werden, wenn also Auftraggeber, Anwender, Auftragnehmer und Anforderungsentwickler gemeinsam die Eigenschaften des Systems festlegen. Eine grob granulare, informale Spezifikation der Benutzerrechte reicht für weite Teile des Systems sicherlich aus und ist auch der erste Schritt zu einer vollständigen Modellierung. Für kritische Systemteile sind jedoch genauere, fein granularere Benutzerrechte besonders im Hinblick auf eine automatische Transformation in die Implementierung notwendig.

Zur Modellierung von Zugriffs- und Ausführungsberechtigungen im Kontext eines objektorientierten, anwendungsfallgetriebenen Entwicklungsprozesses stellen wir in diesem Kapitel einen neuen Ansatz mit einer prädikativen Spezifikation der Benutzerrechte vor. Dazu erweitern wir die Methodenspezifikationen um einen Berechtigungsabschnitt, der die Benutzerrechte eines Akteurs zum Aufruf einer Objektmethode beschreibt.

Nach einer allgemeinen Vorstellung unseres Ansatzes im Abschnitt 4.1 stellen wir in Abschnitt 4.2 klassische Zugriffskontrollmodelle als Basis für unseren Spezifikationsansatz vor. Als syntaktischen und semantischen Rahmen verwenden wir Prädikatenlogik erster Stufe mit einer integrierten Objekt- und Klassenstruktur. Diesen mit einer algebraischen Semantik ausgestatteten Spezifikationsrahmen P-MOS führen wir im Abschnitt 4.3 ein. Die formale Modellierung der Benutzerrechte inklusive einer Repräsentierungsfunktion zur Abbildung von Akteuren im System ist Gegenstand des Abschnitts 4.4. Im Abschnitt 4.5 beschäftigen wir uns mit der formalen Spezifikation von Benutzerrechten mit der Object Constraint Language (OCL) als Spezifikationssprache im Kontext der grafischen Beschreibungssprache UML. Abschnitt 4.6 zeigt Erweiterungen zum vorgestellten Basisansatz und Abschnitt 4.7 gibt einen Ausblick auf die Transformation der in OCL spezifizierten Berechtigungen in die Implementierung. Die Einordnung des Ansatzes in den frühen Phasen eines Entwicklungsprozesses wird in Abschnitt 4.8 beschrieben.

4.1 Einführung

Der Schutz von Daten vor unautorisierten Benutzerzugriffen ist eine grundlegende Anforderung bei der Entwicklung zugriffssicherer Systeme. Anwendungen, wie beispielsweise ERP-Systeme[1] oder Gesundheitsinformationssysteme mit hunderten oder tausenden von Benutzern, die sensible Daten handhaben müssen, enthalten komplizierte Mechanismen zur Rechtemodellierung. Mit der Einführung von Web-Anwendungen hat das zentrale Thema des Schutzes von Daten noch weiter an Bedeutung gewonnen. Je mehr Firmen ihre grundlegenden Geschäftsprozesse ihren externen Partnern öffnen, desto wichtiger wird eine fein granulare Anwendung von Benutzerrechten.

Wie wir in den Kapiteln 6 bis 8 noch zeigen werden, ist der Schutz der Daten eng mit den zwei Sicherheitsaspekten Authentifizierung und Zugriffskontrolle verknüpft. Während sich die Authentifizierung damit beschäftigt, agierende Akteure im System zu identifizieren, betrachtet die Zugriffskontrolle (siehe hierzu Abschnitt 4.2) den Schutz von Informationsquellen.

Mit dem bekannten RBAC-Modell [FCK03, FSG$^+$01, San96] wurde ein geeignetes Paradigma zur Implementierung von Benutzerrechten entwickelt. Trotz des darin eingeführten Rollenparadigmas (vgl. Abschnitt 4.2.2) bleibt die Zugriffskontrolle in realen Anwendungen eine schwierige und komplexe Aufgabe. Die Zugriffskontrolle betrifft in den meisten Fällen verschiedene Schichten, angefangen von der Dialogschicht mit der Benutzerschnittstelle bis hin zur Anwendungsschicht und Datenbankschicht (vgl. Schichtenarchitekturen in [HNS00, Bal96, Bre01]). Darüber hinaus benötigen wir für neu entstehende B2B-Anwendungen[2] neue, berechtigungsbasierte Techniken zur Gewährleistung von Benutzerrechten (vgl. [MFSK97]).

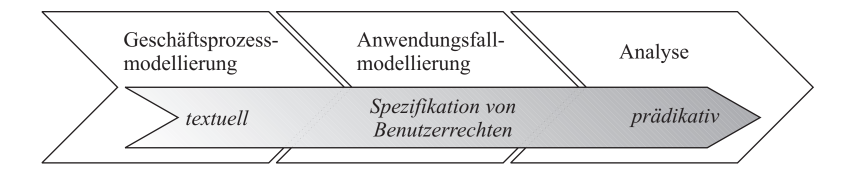


Abbildung 4.1: Beschreibung von Benutzerrechten in der Anforderungsspezifikation

Trotz der Komplexität der Aufgabe wurde bis jetzt eine Rechtemodellierung innerhalb der Analysephase in der Literatur nicht erwähnt. Die Entwicklung von Rechtekonzepten für große Anwendungen, wie beispielsweise in den Bereichen Gesundheit, E-Government oder Wissensmanagement, erfordern sowohl einen schrittweisen Analyseansatz als auch ein implementierungsunabhängiges Analyseframework. Da Benutzerrechte in enger Zusammenarbeit von Auftraggebern, Anwendern, Auftragnehmern und Entwicklern zu entwickeln sind, muss mit der Modellierung der Benutzerrechte in der frühen Phase der Anforderungsspezifikation begonnen werden. Sowohl in der Geschäftsprozessmodellierung, in der Anwendungsfallmodellierung als auch in der Analyse sind die Benutzerrechte zu definieren, zu verfeinern und zu formalisieren. Abbildung 4.1 zeigt den Zusammenhang der Modellierung der Benutzerrechte und den frühen Phasen der Anforderungsspezifikation. Der im Folgenden vorgestellte statische Ansatz zur Spezifikation von Benutzerrechten basiert insbesondere auf drei Fakten.

[1]Enterprise Ressource Planing Systems
[2]Business to Business Applications

Erstens betrachten wir die Modellierung von Benutzerrechten als Aktivität im Kontext einer objektorientierten Entwicklungsmethode, wie beispielsweise im Kontext des Rational Unified Processes [Kru00], des V-Modells [DW99] oder des Catalysis-Ansatzes [DW98]. Dies bedeutet, dass die vorgeschlagene Methode vollständig in den Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme integriert wird. Der vorgestellte Ansatz stellt somit einen Ausschnitt aus einem Prozessmodell zum Security-Engineering [BBH+03] dar.

Zweitens unterstützt die vorgestellte Methode die schrittweise Entwicklung eines Benutzerrechtemodells. Dieses erstreckt sich von informalen textuellen Aussagen bis hin zu einer vollständigen prädikativen Spezifikation. Die implementierungsunabhängige Spezifikation der Benutzerrechte hat eine Vielzahl von Vorteilen. Der bedeutendste Vorteil ist, dass das entwickelte Modell eine kurze und prägnante Darstellung von Benutzerrechten für viele Teile der Implementierung darstellt. Darüber hinaus ermöglicht dieser Ansatz die automatische Übersetzung der vollständigen Benutzerrechtemodelle in ausführbaren Code.

Drittens ist der vorgestellte Ansatz mit einer formalen Semantik in einer algebraischen Theorie ausgestattet. Auf der syntaktischen Ebene erweitern wir die Methodenspezifikationen um einen *Benutzerrechteabschnitt*, der für jeden Akteur (oder für jede Rolle) das Recht zur Ausführung dieser Methode eines Objektes zu einer gegebenen Klasse beschreibt. Da den Akteuren in diesem Ansatz eine bedeutende Rolle bei der Spezifikation der Benutzerberechtigungen zukommt, bezeichnen wir diese Vorgehensweise als *akteurzentriert*. Die Berechtigungen werden durch ein Prädikat erster Ordnung, oder im Kontext der UML mit einem OCL-Ausdruck (vgl. [WK99, OMG03]), über einer integrierten Objektstruktur beschrieben.

Zur internen Darstellung von Akteuren im System wird eine Repräsentierungsfunktion eingeführt. Diese Repräsentierungsfunktion ist eine Abstraktion der Authentifizierungsprozedur der Implementierung und erlaubt Spezifikationen der Art *"Der Anwender hat das Recht, seine eigenen Daten zu sehen"*. Die Semantik der Berechtigungen und der Repräsentierungsfunktion kann direkt in die in [Bre01, Bre04] vorgestellte algebraische Theorie integriert werden.

Verwandte Arbeiten wurden bereits in der Einleitung im Abschnitt 1.4 diskutiert. Der unten vorgestellte Ansatz geht in mehreren Beziehungen über den in der Literatur vorwiegend diskutierten RBAC-Ansatz [FCK03, FSG+01, San96] dahingehend hinaus, dass wir uns mit einem Entwicklungsprozess eines Benutzerrechtemodells im Kontext objektorientierter Modellierungstechniken beschäftigen. Darüber hinaus stellen wir eine erweiterte Ausdrucksmächtigkeit bei der Spezifikation von Benutzerrechten durch beliebige prädikatenlogische Ausdrücke erster Ordnung zur Verfügung. Ein Ansatz mit ähnlicher Ausdrucksmächtigkeit wird in [LBD02, BDL03] vorgestellt, der sich mit dem primären Ziel der Codegenerierung beschäftigt. Davon abgrenzend liegt unser Fokus auf der Entwicklung von Konzepten, die während des gesamten Entwicklungsprozesses angewendet werden können. Ein Schema zur Modellierung von Benutzerrechten im Kontext von Anwendungsfällen wurde erstmals in [FH97a] präsentiert, wovon Grundgedanken übernommen wurden. Mit speziellen Theorien für die Modellierung von Benutzerrechten im Kontext eines Entwicklungsprozesses und der Überprüfung von Rechten in SAP-Anwendungen beschäftigen sich [HJ03, IBM03].

Die formale Spezifikation als Grundlage für die Beschreibung der Benutzerrechte basiert auf der formalen Sprache P-MOS aus [Bre01, Bre04]. Basis zur vorgestellten Akteurmodellierung bildet die Arbeit [BP04].

4.2 Klassische Zugriffskontrollmodelle

Nachdem wir im vorausgehenden Abschnitt die Notwendigkeit eines Benutzerrechtemodells zum Schutz von Daten motiviert haben, stellen wir im Abschnitt 4.2.1 die Grundlagen des Zugriffsmatrixmodells mit seinen zwei Ausprägungen, der benutzerbestimmbaren und der systembestimmbaren Zugriffskontrolle, vor. Darüber hinaus führen wir die Erweiterung dieses Matrixmodells in Form der rollenbasierten Zugriffskontrolle in Abschnitt 4.2.2 ein. Grundlagen der Zugriffskontrolle und Zugriffskontrollmodelle werden unter anderem in [Eck03, Rup02, And01] diskutiert.

Neben diesen beiden Stellvertretern gehören auch das Chinese-Wall-Modell und das Bell-LaPadula-Modell zur Klasse der Zugriffskontrollmodelle. Das *Chinese-Wall-Modell* [BN89] kann deshalb nicht für eine Beschreibung der Zugriffsrechte in der hier benötigten Weise herangezogen werden, da es keine allgemeinen Zugriffsberechtigungen regelt, sondern eine spezielle kommerzielle Strategie verfolgt, die besonders für Anwendungen im Umfeld von Unternehmensberatungen geeignet ist. Dieses Modell verhindert den Informationsfluss zwischen konkurrierenden Systemen. Ohne weiteres ist es jedoch nicht möglich, über die sehr restriktiven, systembestimmbaren Regeln hinausgehend, erlaubte und verbotene Daten- und Systemzugriffe individuell, abhängig von den Anwenderanforderungen, zu modellieren.

Im *Bell-LaPadula-Modell* [BL73] lassen sich auch systembestimmte Regeln einfach überprüfen und effizient implementieren. Dabei schreibt dieses Modell keine Granularität für Objekte und Subjekte vor. Die Rechtefestlegungen, kombiniert mit den restriktiven, systembestimmten Festlegungen des Modells, schränken das Spektrum der zu beschreibenden IT-Systeme jedoch sehr ein, sodass sich dieses Modell ebenenfalls nicht für die Beschreibung von universellen Benutzerrechten eignet.

4.2.1 Zugriffsmatrixmodell

Das Zugriffsmatrixmodell, auch bekannt als Referenzmonitor, ist das einfachste und älteste Sicherheitsmodell. Es bietet die Möglichkeit, Objekte und Subjekte zugeschnitten auf die Anforderungen der jeweilig zu konstruierenden Anwendung festzulegen sowie Zugriffsrechte universell oder objektspezifisch zu modellieren. Bei den Zugriffsmatrixmodellen unterscheiden wir zwei Strategieansätze, die *benutzerbestimmte* und die *systembestimmte* Zugriffskontrolle.

Die benutzerbestimmbare Zugriffskontrolle (engl. *discretionary access control* (DAC)) basiert auf dem Eigentümerprinzip. Dabei ist der Eigentümer für den Schutz der Objekte verantwortlich und die Zugriffsrechte werden auf der Basis einzelner Objekte vergeben oder zurückgenommen. Mit der benutzerbestimmbaren Kontrolle werden somit nur objektbezogene Sicherheitseigenschaften, aber keine systemglobalen Eigenschaften festgelegt. Da bei der Festlegung der individuellen, objektbezogenen Beschränkungen die Abhängigkeiten zwischen Objekten, wie beispielsweise Benutzungs-, Kommunikations- und Kooperationsabhängigkeiten, nicht betrachtet werden, besteht die Gefahr, inkonsistente Strategien zu modellieren.

DAC-Modelle werden üblicherweise mit Zugriffskontrollmatrizen beschrieben. Dabei wird der Schutzzustand eines Systems zum Zeitpunkt t durch eine $|S| \times |O|$-Matrix M_t dargestellt. Dabei gilt:

- Die Spalten der Matrix werden durch die Menge O_t der Objekte und

- die Zeilen der Matrix durch die Menge S_t der Subjekte zum Zeitpunkt t definiert.
- Es gilt: M_t: $|S| \times |O| \rightarrow 2^R$, wobei R die Menge der Zugriffsrechte festlegt (2^R bezeichnet dabei die Potenzmenge der Menge R). Ein Eintrag $M_t(s, o) = \{r_1, \ldots r_n\}$ beschreibt die Menge der Rechte, die das Subjekt s zum Zeitpunkt t an dem Objekt o besitzt.

Für jeden Zeitpunkt t modelliert die Matrix M_t die in dem betreffenden Zustand gültigen Zugriffsrechte der Subjekte an den Objekten des Systems.

Mögliche Zugriffsrechte sind zum Beispiel Lesen, Schreiben oder Ausführen von Prozeduren, Löschen oder Anlegen. Die Granularität der Objekte und Subjekte ist ein wichtiger Parameter bei der Konzeption eines diskreten Zugriffskontrollmodells mit weit reichenden Auswirkungen auf den späteren Betrieb.

Alle Objekte haben einen Eigentümer, der die Rechte an dem Objekt besitzt und verwaltet. Er kann sie an weitere Subjekte ohne vorherige Vermittlung und Genehmigung eines Administrators weitergeben oder auch widerrufen. Die Verwaltung der Rechte und die durch sie beeinflusste Zugriffskontrolle unterliegen somit der Diskretion des Benutzers, was dem Zugriffsmodell auch den Namen gab.

Systembasierte (regelbasierte) Festlegungen (engl. *mandatory access control* (MAC)) spezifizieren systemglobale Eigenschaften, die benutzerbestimmten Rechten übergeordnet sind. Ein Zugriff wird verweigert, wenn es eine verbietende systembestimmte Festlegung gibt, auch wenn der Zugriff durch eine benutzerbestimmbare Strategie erlaubt wäre. Systembestimmte Rechte können in diesem Modell jedoch auch durch benutzerbestimmbare, explizite Berechtigungsverbote weiter eingeschränkt werden. Das MAC-Modell stellt somit eine Erweiterung der benutzerbestimmbaren Zugriffskontrolle dar.

Nach [Eck03] sind die beiden Zugriffsmatrixmodelle für Anwendungsfälle geeignet, in denen Objekte unterschiedlicher Granularität und unterschiedlichen Typs zu schützen sind und im Wesentlichen Integritätsanforderungen stellen. Wegen der objektbezogenen Modellierung sollten die Objekte möglichst wenige Wechselwirkungen beziehen, um inkonsistente Festlegungen zu minimieren. Wegen seiner mangelhaften Skalierbarkeit eignet sich dieser Matrixansatz kaum zur Modellierung einer großen Menge von Subjekten, die sich in hoher Frequenz dynamisch ändert.

Da wir in unserem Ansatz sehr wohl Systeme beschreiben wollen, die mit einer hohen Anzahl von Benutzern umgehen, abstrahieren wir von einzelnen Benutzern zu stellvertretenden Rollen und betrachten im folgenden Abschnitt das hierzu geeignetere Modell der rollenbasierten Zugriffskontrolle.

4.2.2 Rollenbasierte Zugriffskontrolle

Die rollenbasierte Zugriffskontrolle ist ein Zugriffskontrollmodell, welches aufgrund der hohen Flexibilität bei der Beschreibung und Durchsetzung von unternehmensspezifischen Sicherheitspolitiken vor allem in kommerziellen Informationssystemen eingesetzt wird. 1996 wurden rollenbasierte Zugriffsmodelle (RBAC) (engl. *role-based access control*) von Ravi Sandhu [San96] eingeführt.

Im RBAC-Modell findet eine Abstraktion der Aufgaben von Benutzerebene auf Rollenebene statt, d.h., Aufgaben werden an Rollen gebunden und Benutzer werden Rollen zugeordnet.

Gegenüber den bereits vorgestellten Zugriffsmatrixkontrollstrategien werden hier die Zugriffsrechte aufgabenbezogen vergeben, d.h., Zugriffsrechte werden abhängig von und zugeschnitten auf die Aufgaben vergeben, die Subjekte im System durchführen. Subjekte erhalten dann über explizit festzulegende Rollenmitgliedschaften Berechtigungen zur Ausführung von Aufgaben. Rollen sind hier eine Weiterentwicklung des Gruppenkonzepts, bei dem Mengen von Benutzern nach verschiedenen Kriterien zusammengefasst werden.

Das Basismodell eines rollenbasierten Sicherheitsmodells wird in [Eck03] durch ein Tupel $RBAC = (S, O, RL, P, sr, pr, session)$ definiert, wobei

- S die Menge der Benutzer im System,
- O die Menge der zu schützenden Objekte und
- RL die Menge der Rollen ist. Jede Rolle beschreibt eine Aufgabe und damit die Berechtigungen der Rollenmitglieder.
- P ist die Menge der Zugriffsberechtigungen und
- sr, pr und $session$ beschreiben Relationen. Die Abbildung sr modelliert die Rollenmitgliedschaft eines Subjektes $s \in S$,

  $$sr : S \to 2^{RL}.$$

  Falls $sr(s) = \{R_1, \dots R_n\}$ gilt, dann ist das Subjekt s autorisiert in den Rollen $R_1, \dots R_n$ aktiv zu sein. Über die Berechtigungsabbildung

  $$pr : RL \to 2^P$$

  wird jeder Rolle $R \in RL$ die Menge derjenigen Zugriffsrechte zugeordnet, die die Mitglieder der Rolle während ihrer Rollenaktivitäten wahrnehmen dürfen. Die Relation $session \subseteq S \times 2^{RL}$ beschreibt Paare (s, rl), die als Sitzungen bezeichnet werden. Für ein Subjekt $s \in S$ und für eine Sitzung $(s, rl) \in session$ gilt, dass s alle Berechtigungen der Rollen $R \in rl$ besitzt. Sei (s, rl) eine Sitzung, dann sagen wir, dass s aktiv in den Rollen $R \in rl$ ist.

Ein Subjekt kann in einem RBAC-System gleichzeitig in verschiedenen Sitzungen aktiv sein. Es besitzt dann alle Berechtigungen aller Rollen, in denen es aktiv ist, mit Ausnahme derer, die sich gegenseitig ausschließen. Um Berechtigungen wahrnehmen zu können, muss ein Subjekt aktiv in einer Rolle sein, für die die Berechtigung erteilt wurde.

Im Zusammenhang mit großen Informationssystemen, wie sie in Unternehmen und Behörden eingesetzt werden, sind Aufgaben und Zuständigkeiten innerhalb von Abteilungen und Projekten oftmals hierarchisch zu ordnen. Durch die Erweiterung um eine partielle Ordnung auf den Rollen lassen sich mit dem RBAC-Modell auch hierarchische Berechtigungsstrukturen erfassen. Wir erweitern hierzu das gegebene Modell wie folgt:

- Die Relation $rh \subseteq RL \times RL$ ist eine partielle Ordnung auf den Rollen, bezeichnet als Vererbungsrelation, geschrieben als $\geq$, wobei $r_1 \geq r_2$ nur dann gilt, wenn alle Berechtigungen von r_2 auch Berechtigungen von r_1 sind und alle Subjekte von r_1 auch Subjekte von r_2 sind.

- Die Funktion $as(r : RL) \rightarrow 2^S$ ist eine Abbildung der Rolle r auf eine Menge von Subjekten, welche die autorisierten Subjekte liefert.
- Die Funktion $ap(r : RL) \rightarrow 2^P$ ist eine Abbildung der Rolle r auf eine Menge von Berechtigungen, welche die autorisierten Berechtigungen liefert.

Die Vererbung von Benutzerrechten kann jedoch auch problematisch sein, da dadurch ein Subjekt unter Umständen mehr Rechte erhält, als es zur Erfüllung einer Aufgabe benötigt.

Mit einem derartigen RBAC-Modell sind Objekte anwenderspezifisch modellierbar und die Zugriffsberechtigungen für Objekte können somit aufgabenbezogen vergeben werden. Mit der RBAC-Strategie lassen sich die Nachteile der matrixbasierten Zugriffskontrolle aus dem vorausgehenden Abschnitt vermeiden, da sich die Berechtigungsprofile von Rollen nur selten ändern. Auf diese Weise lassen sich Probleme der mangelnden Skalierbarkeit stark reduzieren. RBAC-Modelle eignen sich gut für einen Einsatz in verteilten Systemen.

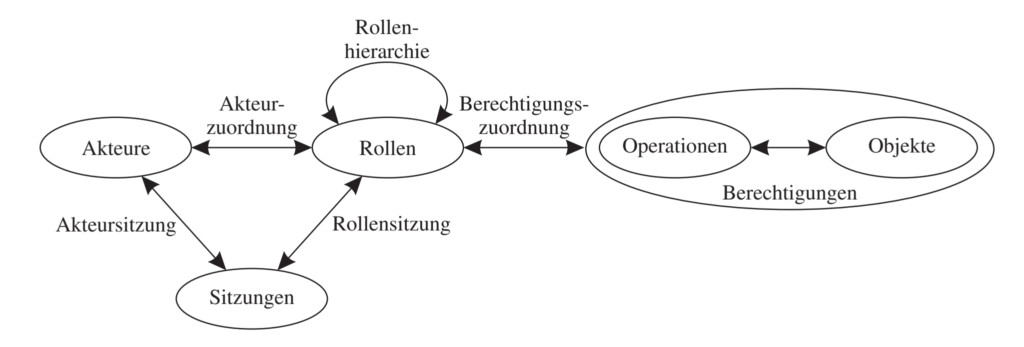


Abbildung 4.2: Schema der rollenbasierten Zugriffskontrolle

Die zu vergebenden Zugriffsrechte können mit diesem Modell objektspezifisch oder universell modelliert werden. Es werden a priori keine Festlegungen hinsichtlich der Menge der zu vergebenden Zugriffsrechte getroffen. Das Modell ermöglicht eine Vergabe der Rechte gemäß des *need-to-know*-Prinzips (vgl. Abschnitt 2.2.3 zu den Sicherheitsprinzipien) und unterstützt eine systematische Aufgabenteilung. Die eingeführte Modellierung einer Rollenhierarchie erlaubt es unmittelbar, existierende Organisations- und Verwaltungsstrukturen von Unternehmen abzubilden, so dass in diesem Umfeld RBAC-Modelle sehr gut bei der Softwareentwicklung eingesetzt werden können.

Abbildung 4.2 gibt nochmals die Zusammenhänge der eingeführten Modellierungskonzepte wieder.

4.3 Der Spezifikationsrahmen P-MOS

Als Spezifikationsrahmen verwenden wir die Spezifikationssprache P-MOS [Bre04, Bre01]. P-MOS ist eine Sprache von Ausdrücken und Prädikaten erster Ordnung (vgl. [Bro98a, RN03, RP97]) zum Navigieren über Objekten, Zuständen und Objektbezügen. Die Ausdrücke und Prädikate in P-MOS beziehen sich auf ein gegebenes Klassendiagramm und sind rein statisch. Insbesondere liefern Ausdrücke in P-MOS einen Wert zurück, sind aber nicht verhaltensändernd. Ausdrücke und Prädikate beziehen sich auf den Zustand einer implizit gegebenen

Menge von Objekten zu einem festen Zeitpunkt. In P-MOS sind die Ausdrücke und Prädikate getypt. Die möglichen Typen stützen sich dabei auf ein gegebenes Typdefinitionsschema einer funktionalen und algebraischen Sprache ab. Komplexe Typausdrücke können sowohl gegebene Grunddatentypen als auch Klassen enthalten. Die syntaktischen Grundkonstrukte der Sprache P-MOS werden nachfolgend beschrieben.

Betrachtet man das Verhältnis der prädikativen Sprache P-MOS und der UML-eigenen Sprache OCL (Object Constraint Language) [San96, WK99], so erkennt man, dass die beiden Sprachen in ihrem Kern syntaktisch als auch konzeptionell gleich sind, vor allem was die Navigation über Attribute und Assoziationen betrifft (vergleiche [Bre01]). P-MOS orientiert sich an algebraischen Ansätzen und deckt mit wenigen Konzepten die volle Ausdrucksstärke der Prädikatenlogik erster Stufe ab. Die OCL orientiert sich mit ihren Konstrukten mehr an Ausführbarkeit und Datenbank-ähnlichem Navigieren in Objektstrukturen. Im Gegensatz dazu enthält die OCL einen stark ausgeprägten Anteil vordefinierter Datentypen mit mächtigen Funktionen und nützlichen syntaktischen Abkürzungen.

In unserem Ansatz dient P-MOS als Zwischensprache für die Entwicklung von Konzepten. Sie liefert die notwendige Semantik für die Einführung der Konzepte. Die OCL verwenden wir als Zielsprache innerhalb unserer Methode bei der Spezifikation von Benutzerrechten.

4.3.1 P-MOS-Ausdrücke

Ein P-MOS-Ausdruck beschreibt eine Navigation in einer durch ein Klassendiagramm induzierten Objektstruktur, die einen Wert abliefert.

Semantisch wird jeder P-MOS-Ausdruck im Kontext einer so genannten Objektumgebung interpretiert, welche eine konkrete Objektstruktur über einem gegebenen Klassendiagramm beschreibt. P-MOS ist eine hybride Sprache, d.h., wir unterscheiden zwischen Basistypen und Klassentypen. Während ein Ausdruck eines Basistyps (wie zum Beispiel *Bool* oder *int*) in einem Wert (*true*, *false*, *0*, *1*, ...) resultiert, liefert ein Klassentyp-Ausdruck eine Referenz auf ein Objekt innerhalb der gegebenen Objektstruktur zurück.

P-MOS-Ausdrücke können mit der Anwendung der folgenden sechs Regeln gebildet werden. Dabei wird angenommen, dass eine Menge von Basisdatentypen (wie *Bool* und *int*) sowie Typkonstruktoren zur Verfügung stehen. Vorausgesetzt wird dabei auch ein Typkonstruktor *Set[_]* zum Aufbau von Mengen über willkürliche Elemente.

1. Basisfunktionen

Sei $f\colon (s_1, \ldots, s_n)\ s$ eine Funktion der Basistypen (wie zum Beispiel *+: (int, int) int* oder *true: () Bool*). Wenn $e_1, \ldots, e_n$ P-MOS-Ausdrücke über den Typen $s_1, \ldots, s_n$ sind, dann ist $f\,(e_1, \ldots, e_n)$ ein P-MOS-Ausdruck vom Typ s.

2. Variablen

Sei X eine Menge von getypten Variablen, dann ist jede Variable x vom Typ s ein P-MOS-Ausdruck vom Typ s.

Die Variable eines Klassentyps steht dabei für einen Objektidentifikator. Sie stellt einen vom Entwickler gegebenen *Objektnamen* dar und dient als Ausgangspunkt für eine Navigation in der zugrunde liegenden Objektstruktur.

Beispiele

klaus : User
setX : Set[Integer]
projektAktivitaeten: Set[Activity]

3. Attribute

Sei *A:T* ein Attribut der Klasse C (*T* ist entweder ein Klassenname oder ein Basistyp) und *e* ein P-MOS-Ausdruck vom Typ *C*. Dann ist *e.A* ein P-MOS-Ausdruck vom Typ *T*.

Beispiele

klaus.name und
mewadis.name

sind P-MOS-Ausdrücke vom Typ *String* aus dem Klassendiagramm in Abbildung 4.3.

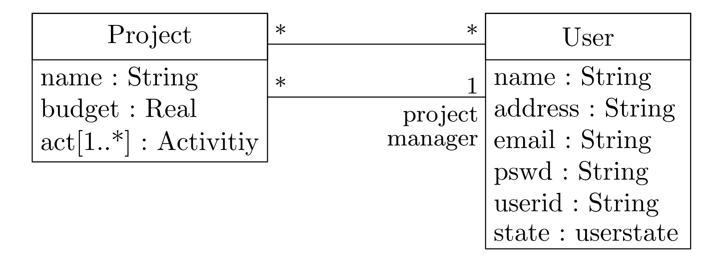


Abbildung 4.3: Klassendiagramm mit Attributen und Assoziationen

Spezialfälle sind Attribute mit Vielfachheiten *A[a..b]:T*. Falls $[a..b] \neq [1..1]$, ist *e.A* ein mengenwertiger Ausdruck, d.h., vom Typ *Set[T]*. Ist $[a..b] = [1..1]$, so ist *e.A* vom Typ *T*.

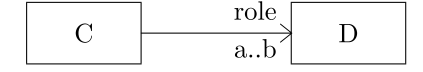
Beispiel

mewadis.act

ist ein P-MOS-Ausdruck vom Typ *Set[Activity]* im Klassendiagramm oben (Abbildung 4.3).

4. Assoziationen

Gegeben sei eine Assoziation in dem folgenden Klassendiagramm. Der P-MOS-Ausdruck *e.role* ist vom Typ *Set[D]* (beziehungsweise *D* im Spezialfall a..b = 1..1), wenn *e* ein Ausdruck vom Typ *C* ist.



Der Rollenname kann auch weggelassen werden. In diesem Spezialfall erhält die Assoziation den vordefinierten Rollennamen *d* (als klein geschrieben Klassennamen). Dies ist allerdings nur möglich, wenn dadurch keine Zweideutigkeiten entstehen (zum Beispiel bei mehrfachen Assoziationen).

Bei bidirektionalen Assoziationen können Ausdrücke durch Navigation in beide Richtungen gebildet werden.

Beispiel
mewadis.projectmanager

ist ein P-MOS-Ausdruck vom Typ *User* bezüglich der in Abbildung 4.3 dargestellten Assoziation *projectmanager*.

5. Generalisierung

Ist e ein P-MOS-Ausdruck vom Typ C und C ist Subklasse von Klasse D (in der transitiven Hülle), dann ist e auch ein P-MOS-Ausdruck vom Typ D. Da jedes Objekt einer Subklasse auch Objekt der Superklasse ist, wird jeder Ausdruck einer Subklasse auch Ausdruck der Superklasse. Diese Eigenschaft wird in vielen Ansätzen auch als *(Subtyp-)Polymorphismus* bezeichnet.

6. Zustandsfunktionen

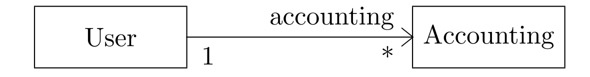
Um die Formulierung von P-MOS-Ausdrücken zu erleichtern, erlaubt P-MOS die Definition von Zustandsfunktionen. Zustandsfunktionen sind Hilfsfunktionen, die wie P-MOS-Ausdrücke eine Navigation in einer gegebenen Objektstruktur in Abhängigkeit von Parametern beschreiben. Im Unterschied zu den Funktionen der Datentypen sind Zustandsfunktionen damit abhängig vom Zustand der Objekte.

Sind nun ***funct*** $f{:}(T_1,\ldots,T_n)\ T$ eine Zustandsfunktion (wobei $T_1,\ldots,T_n,\ T$ entweder Basistypen oder Klassen sind) und $e_1,\ \ldots,\ e_n$ P-MOS-Ausdrücke der Typen $T_1,\ldots,T_n$, dann ist $f(e_1,\ \ldots,\ e_n)$ ein P-MOS-Ausdruck vom Typ T.

Eine Zustandsfunktion beschreibt eine auf n Parametern basierende, generische Navigation in einer gegebenen Objektstruktur. Sie liefert ein Ergebnis vom Typ T als Basistyp oder Objektreferenz. Eigenschaften von Zustandsfunktionen werden durch P-MOS-Prädikate definiert (siehe hierzu Abschnitt 4.3.2).

Abfrageoperationen in Klassendiagrammen beinhalten auch Zustandsfunktionen. Für jede Abfrageoperation $f{:}(x_1{:}T_1,\ldots,x_n{:}T_n){:}T$ in Klasse C definieren wir eine Zustandsfunktion ***funct*** $f{:}(C,T_1,\ldots,T_n)\ T$ und schreiben Ausdrücke $f(e,e_1,\ \ldots,\ e_n)$ als $e.f(e_1,\ \ldots,\ e_n)$ in der gewohnten Weise.

Beispiel
Im Kontext des Klassendiagramms



definieren wir eine Zustandsfunktion, welche die Umkehrrichtung der Assoziation beschreibt:

funct *inhaber (Accounting) User*

Die Form der Prädikate vorwegnehmend, wird die Zustandsfunktion *inhaber* im Beispiel durch das folgende Prädikat spezifiziert.

$$\forall\, ac : Accounting\,.\,\forall\, u : User\,.\; inhaber(ac) = u \Leftrightarrow ac \in u.accounting$$

Referenziert ein Benutzer (User) also eine Buchung (Accounting), dann ist er der Inhaber der Buchung. Die Konsistenz dieser Spezifikation ist dadurch gewährleistet, dass jede Buchung von genau einem Benutzer referenziert wird.

4.3.2 P-MOS-Prädikate

Basierend auf der vorab definierten Struktur der P-MOS-Ausdrücke lassen sich in gewohnter Form Prädikate bilden. Die Struktur der Prädikate wird in Tabelle 4.1 zusammengefasst. Die Quantoren in den Prädikaten beziehen sich auf alle zum gegebenen Zeitpunkt existierenden Objekte.

Tabelle 4.1: P-MOS-Prädikate

P-MOS-Prädikat	**Elementbeschreibung**
$e1 = e2$	$e1$, $e2$ P-MOS Ausdrücke gleichen Typs
$\neg P$, $P1 \vee P2$, $P1 \wedge P2$, $P1 \Rightarrow P2$, $P1 \Leftrightarrow P2$	P, $P1$, $P2$ Prädikate
$\forall x{:}T.P$, $\exists x{:}T.P$	P Prädikat, x Variable und T Typausdruck

Beispiel
Die folgende informal angegebene Invariante, die global im *TimeTool*-System erfüllt sein muss, wird durch das unten angegebene Prädikat formalisiert.

Invariante:
“Zu allen Buchungen muss der zugeordnete Benutzer Mitglied des übergeordneten Projektes sein”

$$\forall\, a : Accounting\,.\, a.user \in a.activity.project.user$$

Wenn alle im Prädikat verwendeten Variablen in Quantoren gebunden sind, sprechen wir von *geschlossenen* Prädikaten. *Freie* Variablen heißen diejenigen, nicht durch Quantoren gebundenen Variablen.

4.4 Formale Modellierung von Benutzerrechten

Der zentrale Begriff für die Erfassung einzelner Benutzer und ihrer Rollen in der Geschäftsprozessmodellierung und in der Anwendungsfallmodellierung ist der des *Akteurs*. Beispielsweise steht der Akteur im Geschäftsprozessmodell für eine Person, oder genauer betrachtet für die Rolle, in der die Person im Geschäftsprozess agiert. Weiterhin können Akteure in der Anwendungsfallmodellierung auch die Rolle von externen Systemen übernehmen.

Die zugrunde liegende Idee für die Modellierung von Benutzerrechten in unserem Ansatz besteht darin, dass Akteure auf den Objekten des Klassendiagramms diverse Berechtigungen besitzen (vgl. Abbildung 4.4). In dieser Hinsicht referenziert unser Benutzerrechtemodell

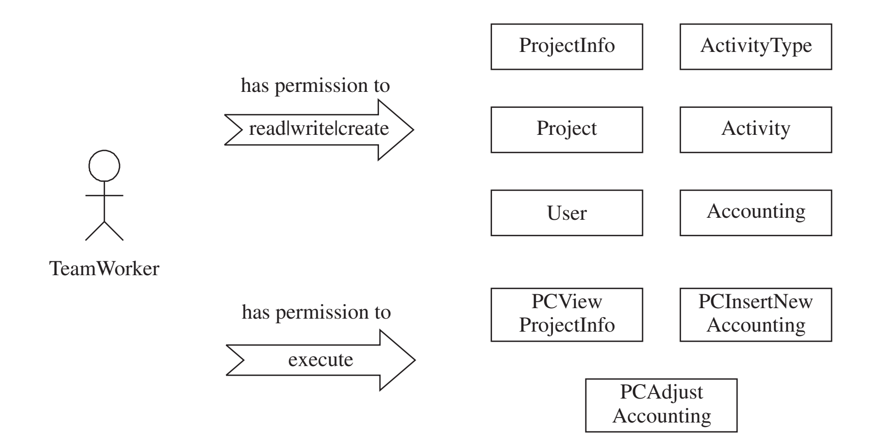


Abbildung 4.4: Akteure besitzen verschiedene Benutzerrechte auf Objekten

sowohl das Modell mit dem Akteur – das Geschäftsprozessmodell oder das Anwendungsfallmodell – als auch das Klassenmodell. Die Trennung des Rollenkonzepts von den Klassen hat den Vorteil, dass die Art und Weise, wie Rollen im System darzustellen sind, nicht innerhalb der Erarbeitung der Anforderungen gelöst werden muss.

In den frühen Phasen der Entwicklung wollen wir die Benutzerrechte oftmals in einer informalen, textuellen Art und Weise spezifizieren. Tabelle 4.2 zeigt einen Ausschnitt für Klassen des Domänenmodells einer derartigen textuellen Spezifikation des Benutzerrechtemodells zu unserer *TimeTool*-Fallstudie. Dieses informale Modell enthält grob granulare Kategorien (R= Read, W = Write, C = Create, E = Execute) für jede Klasse, für Aktivitäten von Geschäftsprozessen oder für Anwendungsfälle. Die textuelle Darstellung charakterisiert die

Tabelle 4.2: Benutzerrechte von Projektmitarbeiter und Projektmanager

actor → ↓ *class*	**Team Worker**		**Project Manager**	
Accounting	R: W/C:	all own accountings (independent of the project) own accountings from released activities	R/W/C:	all accountings of activities of own projects (i.e. where actor is the project manager of)
Activity	R: W/C:	all activities from projects allocated to the actor –	R/W: C:	all activities of own projects –
Project	R: W/C:	all projects –	R: W/C:	all projects –
User	R: W/C:	all users –	R: W/C:	all users –

Objekte einer gegebenen Klasse, welche ein Akteur lesen, schreiben oder erzeugen kann, oder Aktivitäten und Anwendungsfälle, die ein Akteur ausführen kann.

Für große Teile des Systems kann eine derartige Beschreibung von Benutzerrechten ausreichend sein. Für kritische Systemteile benötigen wir jedoch eine fein granularere Weise, um Benutzerrechte ausdrücken zu können. Aus diesem Grund führen wir einen Spezifikationsmechanismus auf der Ebene von Methoden ein. Detailliert wird jede *Methode m* einer *Klasse C* mit einer Berechtigung (engl. *Permission*)

$$perm_{C,m}$$

verbunden, welche spezifiziert, unter welchen Bedingungen ein Akteur Zugang zu einem Methodenaufruf an einem Objekt zu einer gegebenen Klasse besitzt. Im Abschnitt 4.6 stellen wir einen Mechanismus zur Vereinigung von Berechtigungen vor, um eine grob granularere Detailebene zu unterstützen.

In dem skizzierten Rahmenwerk fehlt nun noch eine Verbindung zwischen den Akteuren und ihren internen Klassen. Diese Verbindung wird in den Fällen benötigt, in denen Berechtigungen den Akteur selbst referenzieren, wie in dem Beispiel, bei dem ein *Mitarbeiter Leseberechtigungen zu seinen eigenen Buchungen besitzt.* Um derartige Spezifikationen zu unterstützen, führen wir eine Funktion

$$rolerep$$

ein, die Akteure auf Objekte einer Klasse abbildet. Diese Klasse (in den meisten Fällen eine *User*-Klasse) ist die interne Darstellung der Akteure. Tatsächlich ist diese Repräsentierungsfunktion eine Abstraktion der Authentifizierungsprozedur der Implementierung.

Im Folgenden stellen wir die Repräsentierungsfunktion *rolerep* vor (Abschnitt 4.4.1), gefolgt von dem formalen Mechanismus zur Spezifikation von Berechtigungen auf Methoden (Abschnitt 4.4.2). Die Vererbung von Benutzerberechtigungen ist Gegenstand des Abschnitts 4.4.3.

4.4.1 Die Funktion rolerep

Bereits im letzten Abschnitt haben wir erwähnt, dass die Funktion *rolerep* die Akteure auf ihre interne Darstellung abbildet. Um einen homogenen Spezifikationsrahmen innerhalb P-MOS bereitzustellen, erweitern wir das interne Klassendiagramm um eine Klassenhierarchie zur Darstellung der Akteurrollen.

Hierzu führen wir eine Superklasse *ACActor* zur Modellierung aller Arten von Akteuren ein. Die Subklassen dieser *ACActor*-Klasse bilden die in dem Geschäftsprozessmodell bzw. Anwendungsfallmodell definierten Akteure. Akteurhierarchien des Anwendungsfallmodells werden in eine entsprechende Klassenhierarchie überführt. Zu unserer *TimeTool*-Fallstudie erhalten wir auf diese Weise die Subklassen *ACAdminstrator*, *ACTeamWorker*, *ACProjectManager* und *ACOthers*, wobei *ACProjectManager* eine Subklasse von *ACTeamWorker* ist (siehe Abbildung 4.5).

In dem Modell können die Akteure auch Attribute besitzen. Diese Attribute stellen die Eingaben für die Authentifizierung der Akteure im System dar, wobei Akteure hier Personen oder externe Systeme sein können. In den meisten Fällen sind diese Attribute die Benutzerkennung (engl. *userid*) und ein Passwort. Stellvertretend können aber auch biometrische Daten,

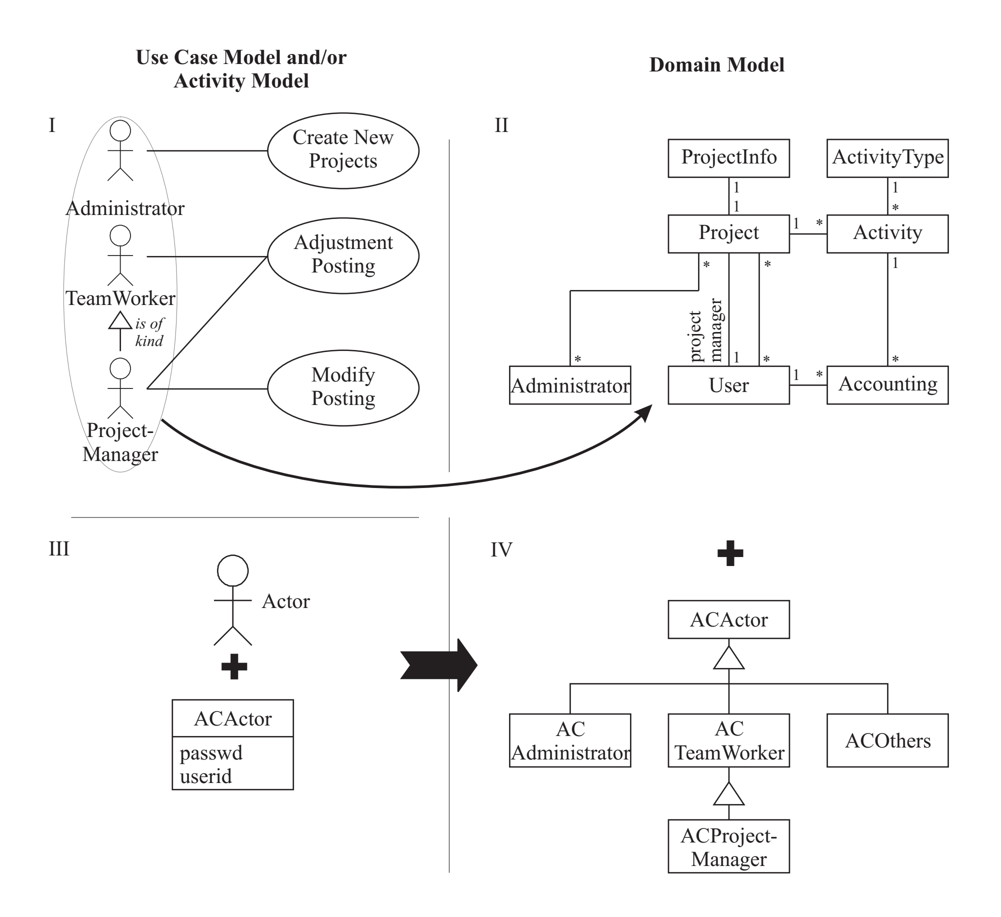


Abbildung 4.5: Erweiterung des Klassendiagramms um Akteurklassen

Ausweise (beispielsweise in der Form von Smartcards) oder gar “keine Zugangsdaten” (falls der Akteur im System anonym bleibt) verwendet werden, wie beispielsweise beim Akteur *ACOthers* der *TimeTool*-Fallstudie.

Formal hat die (zustandsbasierte) Repräsentierungsfunktion *rolerep* die Funktionalität

$$\textbf{\textit{funct}}\; rolerep : (ACActor)\, Object,$$

wobei *Object* die Superklasse aller Klassen ist (wie beispielsweise in der Programmiersprache Java [Fla97]).

Die Funktion *rolerep* ist Teil der Akteurklasse *ACActor* und kann in dieser Superklasse für alle Akteurtypen global, oder aber auch für spezielle Akteurtypen in den einzelnen Subklassen lokal definiert sein. Gleichbedeutend kann die Funktion *rolerep* auch als Anfrageoperation der Funktionalität $rolerep : ()\, Object$ verwendet werden.

Als Beispiele zeigen wir in Abbildung 4.6 die Spezifikation der *rolerep*-Funktionen zu den drei Akteur-Subklassen *ACTeamWorker*, *ACProjectManager* und *ACAdministrator* der Projektverwaltungsfallstudie *TimeTool*.

Wie später noch gezeigt wird, erfolgt die Spezifikation der Akteure im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme. Die Repräsentierungsfunktionen werden bei der Ermittlung der Sicherheitsanwendungsfälle in Anwendungsfällen zur Authentifikation beschrieben.

ACTeamWorker
passwd: String userid: String
rolerep : () Object $\forall\, tw : ACTeamWorker \,.\, \forall\, u : User \,.\, tw.rolerep\,() = u \;\Rightarrow$ $\quad u.state = \sharp active \,\wedge$ $\quad tw.userid = u.userid \;\wedge\; tw.passwd = u.passwd$

ACProjectManager
rolerep : () Object $\forall\, pm : ACProjectManager \,.\, \forall\, u : User \,.\, pm.rolerep\,() = u \;\Rightarrow$ $\quad \exists\, p : Project \,.\, p.projectmanager = u$

ACAdministrator
passwd: String userid: String
rolerep : () Object $\forall\, ad : ACAdministrator \,.\, \forall\, a : Administrator \,.\, ad.rolerep\,() = a \;\Rightarrow$ $\quad ad.userid = a.userid \;\wedge\; ad.passwd = a.passwd$

Abbildung 4.6: Reprästentierungsfunktionen zur TimeTool-Fallstudie

4.4.2 Benutzerberechtigungen

Benutzerberechtigungen (engl. *Permissions*) sind Vorbedingungen zu Methoden, verbunden mit der Semantik, dass die entsprechende Methode nur dann ausgeführt werden kann, wenn der Berechtigungsausdruck (engl. *Permission Expression*) zu Beginn der Methodenausführung zu *true* ausgewertet wird. Da unser Ansatz auf dem *fail-safe default principle* (vgl. Abschnitt 2.2.3 zu den Sicherheitsprinzipien) beruht, ist jede Methodenausführung, die nicht explizit durch eine Berechtigung erlaubt wird, verboten. Jede Methodenberechtigung kann vom aufrufenden Akteur, vom aktuellen Objekt und von den aktuellen Parametern des Methodenaufrufs abhängig sein. Deshalb sind die Benutzerberechtigungen Zustandsfunktionen der Art

$$\textbf{\textit{funct}}\; perm_{\,C,m} : (ACActor, C, T_1, \ldots, T_n)\, Bool$$

wobei C eine Klasse und m eine Methode von C der Form $m\text{-}id : (x_1 : T_1, \ldots, x_n : T_n)\, T$ ist. In dem Spezialfall von Create-Methoden oder Klassenmethoden wird der Parameter C weggelassen.

Die Eigenschaften von Methodenberechtigungen werden durch P-MOS-Prädikate spezifiziert, die Bedingungen über der gegebenen Objektstruktur beschreiben.

Im Folgenden geben wir einige Beispiele zur Spezifikation von Benutzerberechtigungen an. Dabei modellieren wir einige Rechte aus Tabelle 4.2 für Projektmitarbeiter (*team worker*) und Projektleiter (*project manager*) aus der *TimeTool*-Fallstudie. In Abbildung 4.7 ist nochmals

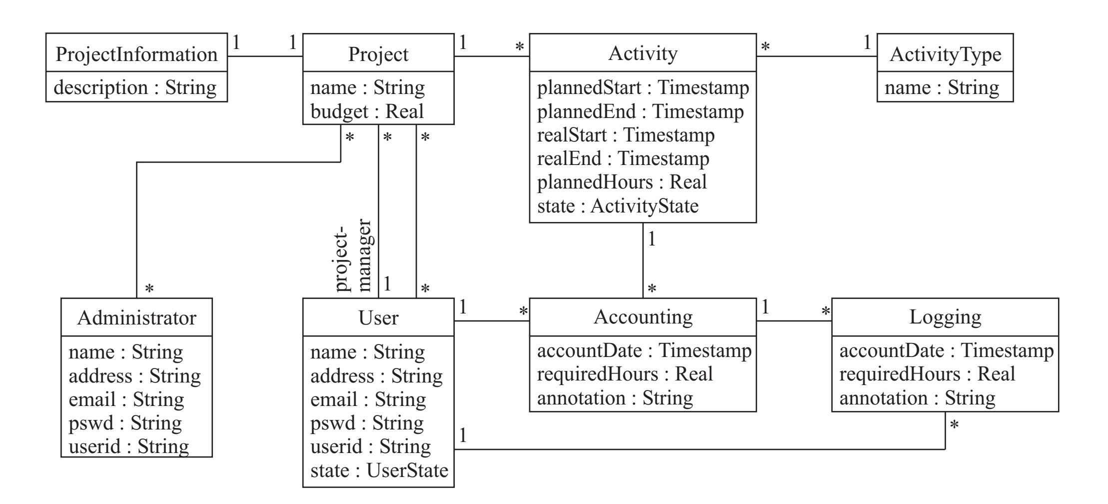


Abbildung 4.7: Das Klassendiagramm zur TimeTool-Fallstudie

das in den Kapiteln 6 bis 8 erarbeitete Klassendiagramm zur *TimeTool*-Fallstudie als Basis für die Navigation in der Objektstruktur dargestellt.

Beispiel 1a

Als erstes Beispiel spezifizieren wir die Berechtigung, dass ein Projektmitarbeiter (*team worker*) alle seine eigenen Buchungen (*Accountings*) lesen kann, unabhängig von dem Projekt, zu dem die Buchungen gehören. Als Beispiel verwenden wir die Methode *getAccountingDate()* als Stellvertreter für eine Lesezugriffsmethode.

$$\begin{aligned}&\forall\, tw : ACTeamWorker\ .\ \forall\, a : Accounting\ .\\&a.user = tw.rolerep\,()\\&\quad\Rightarrow\ perm_{\,Accounting,\,getAccountingDate}(tw, a)\end{aligned}$$

Der Ausdruck legt fest, dass ein User-Objekt, welches mit dem Accounting-Objekt verbunden ist, die interne Repräsentation des gegebenen Projektmitarbeiters (*team worker*) sein muss. Nur wenn dieser Ausdruck zu *true* ausgewertet wird, hat der Projektmitarbeiter Zugriff auf die Methode *getAccountingDate* der Klasse *Accounting*.

Beispiel 1b

Als zweites Beispiel betrachten wir die Berechtigung, dass ein Projektleiter (*project manager*) alle Buchungen zu eigenen Projekten lesen kann. Wir spezifizieren hier erneut die Benutzerberechtigung für die Methode *getAccountingDate*.

$$\begin{aligned}&\forall\, pm : ACProjectManager\ .\ \forall\, a : Accounting\ .\\&a.activity.project.projectmanager = pm.rolerep\,()\\&\quad\Rightarrow\ perm_{\,Accounting,\,getAccountingDate}(pm, a)\end{aligned}$$

Beispiel 2

Eine weitere Berechtigung für den Projektmitarbeiter (*team worker*) spezifiziert, dass er nur Buchungen zu freigegebenen Aktivitäten schreiben kann. Aktivitäten sind mit einem Zustandsattribut gekennzeichnet, das auf *freigegeben* oder *eingefroren* gesetzt wird. Auf diese Weise kann verhindert werden, dass Buchungsobjekte zu abgeschlossenen Aktivitäten im Nachhinein verändert werden. Das Berechtigungsprädikat für die stellvertretende Schreibzugriffsmethode *writeAccountingDate* sieht folgendermaßen aus:

$$\begin{aligned}
&\forall\, tw : ACTeamWorker\ .\ \forall\, a : Accounting\ .\\
&a.user = tw.rolerep\,() \wedge a.activity.state = \sharp released\\
&\quad \Rightarrow\ perm_{\,Accounting,\, writeAccountingDate}(tw, a)
\end{aligned}$$

Beispiel 3

Als letztes Beispiel betrachten wir die Berechtigung zu einer Create-Methode. Der Projektleiter (*project manager*) kann nur Buchungen zu eigenen Projekten erzeugen. Hierbei nehmen wir an, dass die Create-Methode nur die Aktivität und den User, auf den das Accounting-Objekt referenzieren soll, als Parameter übergeben bekommt.

$$\begin{aligned}
&\forall\, pm : ACProjectManager\ .\ \forall\, ac : Activity\ .\ \forall\, u : User\ .\\
&ac.project.projectmanager = pm.rolerep\,()\\
&\quad \Rightarrow\ perm_{\,Accounting,\, create}(pm, ac, u)
\end{aligned}$$

4.4.3 Vererbung von Benutzerberechtigungen

Bei den Benutzerberechtigungen gibt es zwei Möglichkeiten der Vererbung, die sich aus der Vererbung von Klassen im objektorientierten Paradigma und aus der eingeführten Spezialisierung von Akteuren ergeben.

Die erste Vererbungsart bezieht sich auf die Akteurhierarchie (siehe hierzu den Quadrant IV von Abbildung 4.5). In unserem Beispiel ist der Projektleiter (*project manager*) eine spezielle Art des Projektmitarbeiters (*team worker*). Deshalb besitzt ein Projektleiter neben seinen eigenen Berechtigungsaxiomen auch die des Projektmitarbeiters. Beispielsweise besitzt der Projektmanager bezüglich der Berechtigungen aus den Beispielen 1 und 2 in Abschnitt 4.4.2 den Lesezugriff sowohl zu eigenen Buchungen (unabhängig vom Projekt), als auch zu allen Buchungen von eigenen Projekten (unabhängig vom User).

Die zweite Vererbungsart beschäftigt sich mit der Vererbung im Domänenmodell, dessen Objekte mit den Benutzerberechtigungen zu schützen sind. Genauso wie oben erben Subklassen-Methoden die Berechtigungen aus ihren Superklassen. Zusätzlich erlauben wir, dass Berechtigungen in den Superklassen nicht spezifiziert werden und dass dann gegebenenfalls die Berechtigungen in den Subklassen spezifiziert sind.

4.5 Spezifikation von Benutzerrechten in OCL

Im Hinblick auf eine Werkzeugunterstützung zur Modellierung von Benutzerrechten und anlässlich von Implementierungsaspekten ist die formale Spezifikation zur Anwendung aus dem formalen Rahmenwerk in eine Spezifikationssprache im Kontext einer grafischen Beschreibungssprache, wie beispielsweise UML, zu transformieren.

Wir zeigen im Folgenden die Realisierung unserer Konzepte in der Object Constraint Language (OCL) [WK99, OMG03], welche Bestandteil der grafischen Beschreibungssprache UML ist. Wir erweitern hierzu den Spezifikationsabschnitt einer Methode für Vor- und Nachbedingungen um einen zusätzlichen *Berechtigungsabschnitt* (engl. *permission section*). Darin beschreiben wir die Benutzerberechtigungen durch einen OCL-Ausdruck unter Verwendung der Variablen der Methode, des aktuellen Objekts und des aktuellen Akteurs, der ebenfalls wie ein Parameter behandelt wird.

Während Vor- und Nachbedingungen mehr ein Konstrukt des Entwurfs für die abstrakte Sicht auf Klassen und Operationen sind, ist ein Design und eine Implementierung dahin gerichtet, dass diese Bedingungen a priori eingehalten werden und somit zumeist zu *true* evaluieren. Hingegen trifft diese Annahme für Berechtigungsspezifikationen unseres Ansatzes nicht immer zu. In vielen Fällen versuchen Benutzer, die nicht die notwendigen Benutzerrechte besitzen, geschützte Operationen auszuführen.

Die Benutzerrechtespezifikationen unterscheiden sich von Vor- und Nachbedingungen darin, dass sie einen berechneten boolschen Wert zurückgeben. Wird *false* zurückgegeben, schlägt eine weitere Abarbeitung der Methode fehl, was in den gegebenen Fällen auch das Ziel sein soll. Zur Verdeutlichung dieser Eigenheit der Benutzerrechtespezifikation verwenden wir bei der Beschreibung der Berechtigungen eine differenzierte Schreibweise. Mit dem Symbol "`:=`" zwischen der linken und der rechten Seite heben wir hervor, dass von dem Ausdruck der rechten Seite nicht selbstverständlich der boolsche Wert *true* erwartet wird. Das Ergebnis ist also different, während bei Vor- und Nachbedingungen stets der Wert *true* erwartet wird und dies mit einem einfachen "`:`" zwischen linker und rechter Seite des Bedingungsausdrucks dargestellt wird.

Die Repräsentierungsfunktion *rolerep* wird hier wieder als Anfrageoperation der Akteurhierarchie verwendet, wie dies im Abschnitt 4.4.1 eingeführt wurde und wie die Funktion *rolerep* in den Beispielen in Abschnitt 4.4.2 bereits verwendet wurde.

Beispiel 1a

Ein Projektmitarbeiter (*team worker*) kann alle seine eigenen Buchungen lesen, unabhängig von dem Projekt, zu dem die Buchungen gehören (veranschaulicht an der Berechtigung zur Methode *getAccountingDate*).

```
context Accounting :: getAccountingDate()
   perm (act : ACTeamWorker) :=
      self.user = act.rolerep()
```

Beispiel 1b

Ein Projektleiter (*project manager*) kann alle Buchungen zu eigenen Projekten lesen (veranschaulicht an *getAccountingDate*).

```
context Accounting :: getAccountingDate()
   perm (act : ACProjectManager) :=
      self.activity.project.projectmanager = act.rolerep()
```

Beispiel 2

Ein Projektmitarbeiter (*team worker*) kann nur eigene Buchungen zu freigegebenen Aktivitäten schreiben (veranschaulicht an *writeAccountingDate*).

```
context Accounting :: writeAccountingDate()
   perm (act : ACTeamWorker) :=
      self.user = act.rolerep() and
      self.activity.state = ActivityState::released
```

Beispiel 3

Der Projektleiter (*project manager*) kann nur Buchungen zu Aktivitäten von eigenen Projekten erzeugen, d.h., zu Aktivitäten von Projekten, bei denen er Projektleiter ist.

```
context Accounting :: create(a : Activity, u : User)
   perm (act : ACProjectManager) :=
      a.project.projectmanager = act.rolerep()
```

Bezüglich der Semantik von Berechtigungen auf Methoden muss betont werden, dass das dem Akteur zugeordnete Objekt *nicht* von sich aus eine Methode aufruft, sondern die Rolle initiiert den Aufruf einer Methode von außen, d.h., außerhalb des Systems. Ein solcher Methodenaufruf kann auch über eine Kette von Methodenaufrufen erfolgen. Bei der Implementierung kann der aufrufende Akteur entweder als weiterer Methodenparameter behandelt werden oder aber, wie in J2EE, durch eine Infrastruktur für Methodenaufrufe.

4.6 Erweiterungen

Wie wir im Abschnitt 4.4 erklärt haben, ist unsere Basissicht bei der prädikativen Spezifikation der Benutzerrechte die, dass ein Akteur Berechtigungen zum Ausführen von Objektmethode hat. Dieses fein granulare Paradigma stellt uns ein Maximum an Flexibilität für Spezifikationen jeglicher Art von Benutzerbedingungen in allen Phasen der Entwicklung zur Verfügung. Jedoch benötigen wir für die praktische Anwendung einen Integrationsmechanismus zur Unterstützung von grob granulareren Spezifikationen. Deshalb führen wir den Begriff der *Methodenkategorie* ein. Eine Methodenkategorie CAT ist eine Menge von Methoden

$$CAT \subseteq METH,$$

wobei $METH$ die Menge aller Methoden im System ist. Die Methoden sind dabei durch die Klassen sortiert, zu denen sie gehören. Kategorien können eine Menge von Methoden einer einzelnen Klasse oder eine Menge von Methoden mehrerer Klassen sein. Falls eine Kategorie nur aus Methoden einer einzelnen Klasse besteht, verwenden wir den Klassennamen als Index

der Kategorie. Darüber hinaus können Kategorien auch andere Kategorien mit einschließen, sodass Kategorien hierarchisch strukturiert werden können. Wir definieren grob granulare Berechtigungen

$$perm_{Cat_C} : (ACActor, C)\, Bool$$

für jede Kategorie Cat_C, welche Methoden der Klasse C enthält. Kategorieberechtigungen enthalten Methodenberechtigungen in bekannter Weise

$$\begin{aligned}&\forall\, a : ACActor,\ o : C,\ a_1 : T_1, \ldots, a_n : T_n\ .\\&perm_{Cat_C}(a, o) \Rightarrow perm_{C,m}(a, o, a_1, \ldots, a_n)\end{aligned}$$

für alle Methoden m von Klasse C in der Kategorie Cat_C. Diese Kategorieberechtigungen können nur vom Akteur und dem aktuellen Objekt abhängen.

Falls eine Kategorie Cat Methoden von verschiedenen Klassen (und Create-Methoden) umfasst, definieren wir Berechtigungen

$$perm_{Cat} : (ACActor)\, Bool$$

und induzieren die folgenden Methodenberechtigungen für alle Methoden m von Klassen C in Cat:

$$\begin{aligned}&\forall\, a : ACActor,\ o : C,\ a_1 : T_1, \ldots, a_n : T_n\ .\\&perm_{Cat}(a) \Rightarrow perm_{C,m}(a, o, a_1, \ldots, a_n)\end{aligned}$$

Kategorieberechtigungen dieser Art hängen nur von dem Akteur ab, der den Methodenaufruf ausführt. Bezüglich der Anwendung dieses Konzepts stellen wir eine Menge von vordefinierten Kategorien zur Verfügung:

$READ_C$	Kategorie über allen Methoden, die Attribute der Klasse C lesen
$UPDATE_C$	Kategorie über allen Methoden, die Attribute der Klasse C ändern
$CREATE_C$	Kategorie über allen Create-Methoden der Klasse C

Weiterhin definieren wir *get-* und *set-Methoden* zu Attributen als vordefinierte Elemente der $READ_C$- bzw. $UPDATE_C$-Kategorien. Dabei ist zu betonen, dass es Methoden geben kann, die sowohl zur $READ_C$- als auch zur $UPDATE_C$-Kategorie gehören.

Als Beispiel können wir die informalen Berechtigungen aus Tabelle 4.2 direkt in entsprechende Berechtigungen mit Methodenkategorien transformieren. Als Beispiel für eine Kategorieberechtigung transformieren wir die folgende informale Berechtigung: *Ein Projektmitarbeiter kann alle seine eigenen Buchungen lesen, unabhängig vom Projekt.* Die Kategorieberechtigung hierzu lautet:

$$\begin{aligned}&\forall\, tw : ACTeamWorker\ .\ \forall\, a : Accounting\ .\\&a.user = tw.rolerep\,()\\&\quad \Rightarrow perm_{READ_{Accounting}}(ac, a)\end{aligned}$$

Die Menge der vordefinierten Methodenkategorien kann durch benutzerdefinierte Kategorien ergänzt oder ersetzt werden. Dies ist ratsam, falls ein Teil des Klassendiagramms mit derselben Art von Berechtigungen ausgestattet ist. Ein Beispiel hierzu ist ein allgemeines Leserecht *true*. Ein weiterer typischer Fall zur Anwendung von mehr fein granularen Kategorien ist beispielsweise die Unterteilung von Personendaten. Die Attribute einer Klasse *Person* werden in sicherheitskritische (Gehalte, etc.) und unkritische Daten (Name, Adresse, etc.) unterteilt.

4.7 Ausblick auf Codegenerierung

In unserem Ansatz zur akteurzentrierten Modellierung von Benutzerrechten verfolgen wir das Ziel einer von der Implementierung losgelösten Benutzerrechtemodellierung. Dies bedeutet jedoch nicht, dass die Benutzerrechte manuell innerhalb der Implementierung codiert werden müssen. Wie in Abschnitt 4.5 gezeigt wurde, lassen sich unsere Berechtigungsspezifikationen in die Spezifikationssprache OCL, die im Umfeld der UML angewendet wird, überführen. Basierend auf derartigen Berechtigungen in OCL ist hier eine Toolunterstützung angedacht, welche die in einem grafischen Beschreibungstool in OCL annotierten Berechtigungen in ausführbare Methoden überführt. Da die Entwicklung einer derartigen Werkzeugunterstützung der vorgestellten Konzepte erst begonnen wurde und diese auch nicht Fokus dieser Arbeit ist, zeigen wir im Folgenden als Beispiel nur die manuelle Umsetzung der in den Abschnitten 4.4.2 und 4.5 gezeigten Beispiele 1a und 1b als Beispiel für die automatische Generierung.

```
// Allgemeine Berechtigungsüberprüfung
boolean Accounting::getAccountingDatePermission (act : ACActor) {

  // Es liegt keine Berechtigung vor → False
  return false;
}

// Berechtigungsüberprüfung für den Projektmitarbeiter
boolean Accounting::getAccountingDatePermission (act : ACTeamWorker) {

  // Abprüfen der Bedingung über der Objektstruktur
  return this.user == act.rolerep();
}

// Berechtigungsüberprüfung für den Projektleiter
boolean Accounting::getAccoutingDatePermission (act : ACProjectManager) {

  // Abprüfen der Berechtigung für den Projektmitarbeiter
  return getAccountingDatePermission( (ACTeamWorker)act );

  // Abprüfen der Bedingung über der Objektstruktur
  return this.activity.project.projectmanager == act.rolerep();
}

// Die Methode getAccountingDate
Timestamp Accounting::getAccountingDate (act : ACActor) {

  // Überprüfen der Berechtigung
  if ( !getAccountingDatePermission(act) ) return 0;

  // Berechtigung wird gewährt
  return this.accountDate;
}
```

Die Berechtigungsüberprüfung wurde hier in eigene Methoden ausgelagert, die bei der Abarbeitung der Methode *getAccountingDate* aufgerufen werden muss. Anhand der überlade-

nen Berechtigungsmethoden kann abhängig vom übergebenen Akteur die geeignete Berechtigungsüberprüfung ausgewählt werden (hier die *getAccountingDatePermission* für den Projektmitarbeiter oder für den Projektleiter). Die Berechtigungsfunktion für den Projektleiter weist hierbei eine Besonderheit auf. Da ein Projektleiter eine besondere Form des Projektmanagers ist, treffen nach der in Abschnitt 4.4.3 vorgestellten Vererbung von Benutzerrechten die Benutzerrechte des Projektmitarbeiters auch für den Projektleiter zu. Es ist in diesem Zusammenhang zu überprüfen, inwiefern Zugriffe aus der Mitarbeiterrolle des Projektleiters seine Benutzerrechte erweitern.

Nach dem *fail-safe default principle* (vgl. Abschnitt 4.4.2 und Abschnitt 2.2.3 zu den Sicherheitsprinzipien) wird jeder Zugriff, der nicht explizit erlaubt ist, nicht gewährt, was sich in der Umsetzung dadurch zeigt, dass nur die "wahren" Berechtigungsausdrücke spezifiziert werden. Eine Ablehnung des Zugriffs kann nun durch die Nichterfüllung eines derartigen Ausdrucks auftreten oder durch das Fehlen einer Berechtigungsspezifikation. In diesem Fall wird implizit die *getAccountingDatePermission*-Methode der Basisklasse *ACActor* aufgerufen, die *false* zurückliefert. Soll beispielsweise der Zugriff auf eine Objektmethode für alle Akteure ermöglicht werden, so kann in der Basisklasse die globale Benutzerberechtigung *true* spezifiziert und die Zugriffsmethoden in den Subklassen können weggelassen werden.

4.8 Einordnung im Entwicklungsprozess

Abbildung 4.8 zeigt zusammenfassend die Entwicklung unseres Benutzerrechtemodells im Kontext einer objektorientierten, anwendungsfallbasierten Entwicklung der Anforderungsspe-

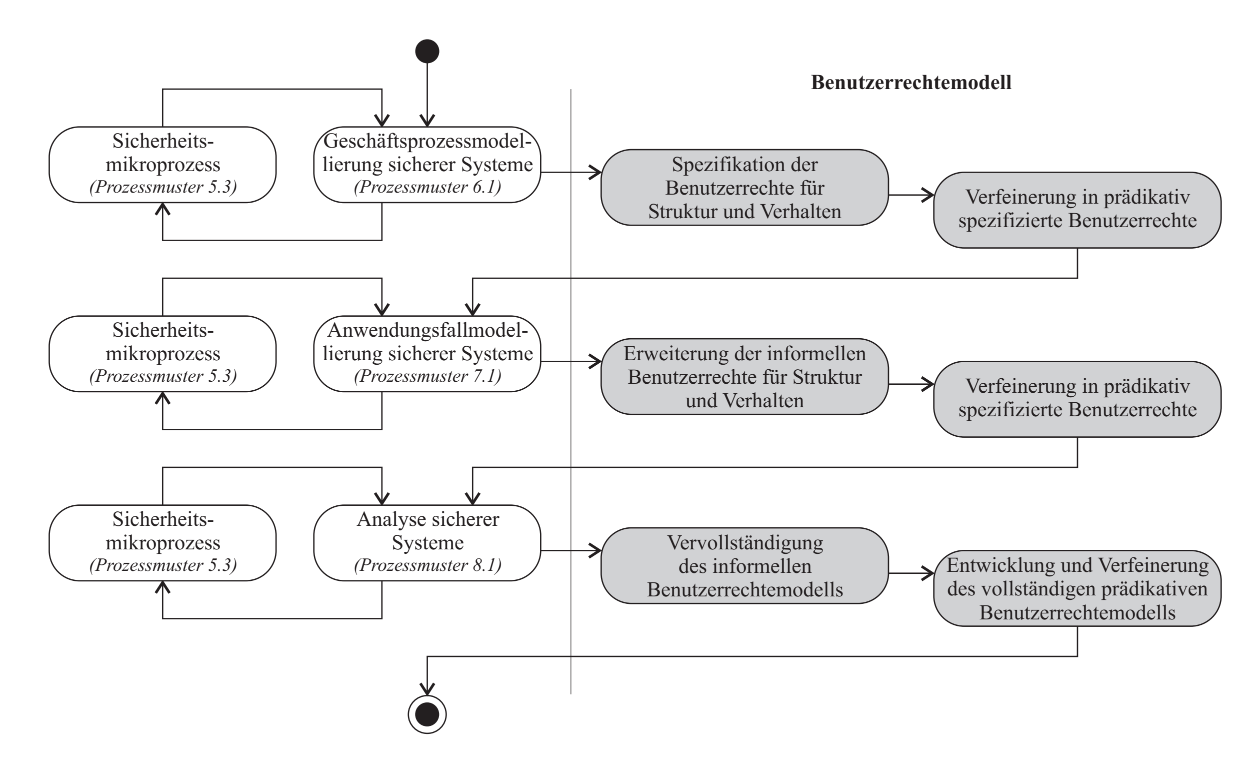


Abbildung 4.8: Modellierung der Benutzerrechte innerhalb der Anforderungsspezifikation

zifikation. Die einzelnen Aktivitäten sind detailliert in den Kapiteln zur Geschäftsprozessmodellierung (Kapitel 6), zur Anwendungsfallmodellierung (Kapitel 7) und zur Analyse (Kapitel 8) zugriffssicherer Systeme beschrieben.

Wie in Abbildung 4.1 gezeigt wurde, erstreckt sich die Modellierung über alle drei Prozessabschnitte und die zunächst informale Spezifikation wird schrittweise in eine prägnante, formale Spezifikation überführt.

Da in den einzelnen Prozessabschnitten das Domänenmodell iterativ weiter entwickelt wird, muss das Benutzerrechtemodell in jedem Abschnitt der Anforderungsspezifikation ergänzt werden.

4.9 Zusammenfassung

In diesem Kapitel haben wir einen formalen Spezifikationsrahmen zur Modellierung von Benutzerrechten eingeführt. Der Ansatz ist dahingehend neu, dass die Benutzerrechte in einer Berechtigungsmatrix vorab textuell spezifiziert und dann schrittweise prädikativ formalisiert werden. Für eine durchgehende und sukzessive Entwicklung ist es entscheidend, dass bereits ab der Phase der Geschäftsprozessmodellierung Sicherheitsaspekte und damit auch Benutzerrechte betrachtet werden, denn zu diesem Zeitpunkt sitzen Entwickler, Anwender, Auftragnehmer und Auftraggeber bei der Festlegung der Anforderungen zusammen und es können bereits sicherheitsrelevante Verhaltensweisen, wie beispielsweise die manuelle Handhabung von Mitarbeiterunterlagen, protokolliert werden.

Neben diesem bereits erwähnten Vorteil der durchgängigen und sukzessiven Entwicklung hat dieser implementierungsunabhängige Ansatz eine Reihe weiterer Vorteile. So kann mit der Modellierung bereits begonnen werden, wenn das Domänenmodell zum Anwendungskern vorliegt. Im Gegensatz zu Ansätzen, in denen die Rechte in den Klassen des Klassendiagramms modelliert werden, ist unser Ansatz mit einer separaten Modellierung unabhängig von kleineren Modifikationen an Assoziationen, Attributen oder Operationen. Da für systemkritische Teile eine textuelle, grob granulare Spezifikation von Benutzerrechten nicht ausreicht, können wir mit unserem Ansatz die vorab informal spezifizierten Rechte in eine fein granularere prädikative Rechtespezifikation transformieren, die die volle Ausdrucksmächtigkeit der Prädikatenlogik erster Stufe unterstützt. Benutzerrechte können damit in beliebiger Granularität auf der Basis von Methodenberechtigungen über einer gegebenen Objektstruktur ausgedrückt werden. Basis für diese Art von Berechtigungsausdrücken ist die mit einer formalen Semantik ausgestattete algebraische Sprache P-MOS. Sie erlaubt uns eine Beschreibung von Berechtigungsausdrücken über Objekten, Zuständen und Objektbezügen von einer gegebenen Menge von Objekten zu einem festen Zeitpunkt.

Besondere Bedeutung kommt bei einer derartigen Benutzerrechtebeschreibung den Akteuren zu. Durch eine zugrunde liegende rollenbasierte Rechtemodellierung werden die Benutzerberechtigungen nicht einzelnen Personen, sondern Rollen zugeordnet. Im Grunde werden dann jeder Person eine oder mehrere Rollen zugeordnet und jede Rolle wird mit einer Menge von Berechtigungen verbunden. Die Akteure können dann durch eine eigens definierte Repräsentierungsfunktion auf die innere Struktur der Anwender abgebildet werden.

Für eine verbesserte Anwendung des Ansatzes haben wir Methodenkategorien eingeführt,

sodass neben den Subjekten (die Akteure) auch die zu schützenden Objektmethoden gruppiert und mit vordefinierten Benutzerrechten ausgestattet werden können.

Unsere Modellierung ist auf eine objektorientierte Vorgehensweise ausgerichtet, die in weiten Teilen die grafische Beschreibungssprache UML verwendet. Im Kontext dieser Sprache findet sich die prädikative Spezifikationssprache OCL wieder, sodass wir unsere prädikatenlogischen Berechtigungsausdrücke direkt in diese Sprache transformieren können. Dies ermöglicht uns die Spezifikation der Benutzerrechte anlag zu Vor- und Nachbedingungen sowie Invarianten unter Verwendung der Variablen der Methoden, des aktuellen Objektes und des aktuellen Akteurs.

Neben der Vorstellung des formalen Rahmenwerks für Benutzerberechtigungen mit der Transformation nach OCL wurde in diesem Kapitel auch ein Ausblick auf eine Transformation in die Implementierung gegeben. Die Berechtigungsspezifikationen in Form von P-MOS-Prädikaten und OCL-Bedingungen sowie die Codegenerierung wurde exemplarisch an der *Time-Tool*-Fallstudie gezeigt.

5 Ein Prozess zur Entwicklung von Anforderungsspezifikationen für zugriffssichere Systeme

Für die Einführung eines Vorgehensmodells in den Phasen der Anforderungsspezifikation für zugriffssichere Systeme sind Prozessabläufe zu definieren und Beschreibungstechniken zu erweitern. Zur Beschreibung von Sicherheitseigenschaften sind die Produktartefakte anzupassen und Neue hinzufügen, beispielsweise zur Spezifikation von Benutzerrechten.

In Abschnitt 5.1 stellen wir zunächst existierende Vorgehensmodelle und Vorgehensbeschreibungen zur Entwicklung sicherer Systeme vor. Für die Definition von Prozessanforderungen, für die Eingliederung der Anforderungsphase in ein Vorgehensmodell und für die Definition einer Anforderungsphase mit Teilabschnitten untersuchen wir diese heutigen Vorgehensweisen bezüglich Vorteile und Eignung zur Entwicklung von zugriffssicheren Systemen.

Die Prozessanforderungen zur Entwicklung zugriffssicherer Systeme stellen wir detailliert in Abschnitt 5.2 dar. Rückblickend bewerten wir die existierenden Vorgehensmodelle und Vorgehensbeschreibungen bezüglich dieser Prozessanforderungen.

In Abschnitt 5.3 wird das eigentliche Vorgehen der Anforderungsspezifikation zugriffssicherer Systeme in Form eines Prozessmusters vorgestellt. Weitere Prozessmuster zeigen die Eingliederung dieser Phase in einen objektorientierten Softwarelebenszyklus und einen Sicherheitsmikroprozess, welcher zum Zwecke der Modellierung von Aspekten der Zugriffssicherheit innerhalb der Anforderungsspezifikation mehrfach zur Ausführung kommt.

Eine Übersicht über die Produktartefakte der Anforderungsspezifikation zugriffssicherer Systeme wird in Kapitel 5.4 gegeben, wobei Produktartefakte aus bekannten objektorientierten Vorgehensmodellen um Aspekte der Zugriffssicherheit ergänzt und für die Modellierung der Zugriffssicherheit spezielle Produktartefakte eingeführt werden. Die Erweiterungen der Produktartefakte werden detailliert in den Prozessabschnitten zur Entwicklung zugriffssicherer Systeme vorgestellt, zur Geschäftsprozessmodellierung in Kapitel 6, zur Anwendungsfallmodellierung in Kapitel 7 und zur Analyse in Kapitel 8. Neben den Ergänzungen an den Produktartefakten wird auch ein Überblick über die Abhängigkeiten der Artefakte gegeben.

Die Abdeckung der Prozessanforderungen des eingeführten Vorgehens ist Gegenstand des Abschnitts 5.5. Die Abdeckung der in Kapitel 5.2 vorgestellten Prozessanforderungen wird am vorgestellten Vorgehen überprüft.

5.1 Existierende Vorgehensmodelle

Die Notwendigkeit, sichere Software nach vorgegebenen Vorgehensmodellen zu entwickeln, wurde nach dem Scheitern von Security-Projekten analog zu herkömmlichen Softwarepro-

jekten erkannt. Erste Methoden zur Entwicklung zugriffssicherer Systeme wurden durch eine Erweiterung von klassischen Vorgehensmodellen geschaffen. Da die klassischen Modelle für die Adaption von Sicherheitseigenschaften kaum offen sind, konnten die notwendigen Änderungen nicht im erforderlichen Umfang eingebracht werden. Diese ersten, um Sicherheitsaspekte erweiterten Vorgehensmodelle enthalten jedoch inhärent eine Reihe von Aspekten, die in ein Vorgehensmodell zur Entwicklung zugriffssicherer Systeme zu integrieren sind. Die Analyse von Stärken und Schwächen existierender Vorgehensmodelle zur Entwicklung sicherer Systeme ist Gegenstand dieses Abschnitts.

5.1.1 Ein phasenstrukturiertes Vorgehensmodell

Da dedizierte Methoden zur Konstruktion sicherer IT-Systeme im Sinne eines systematischen Security Engineerings bislang kaum entwickelt wurden, stellt [Eck03] eine Erweiterung eines phasenstrukturierten Vorgehens mit Sicherheitsaspekten vor. Abbildung 5.1 fasst die Phasen zusammen, die bei einer methodischen Konstruktion sicherer Systeme zu durchlaufen sind.

Aufbauend auf einem Pflichtenheft, in dem die funktionalen Eigenschaften des Systems, seine Einsatzumgebung, der Verwendungszweck, die vorhandenen und einzusetzenden Systemkomponenten und -dienste sowie die Leistungs-, Zuverlässigkeits- und Sicherheitsanforderungen festgehalten sind, wird hier die Entwicklung mit einer Bedrohungsanalyse fortgesetzt. In der Bedrohungsanalyse sind die Gefährdungsbereiche sowie die potenziellen organisatorischen, technischen und benutzerbedingten Ursachen für Bedrohungen systematisch zu entwickeln.

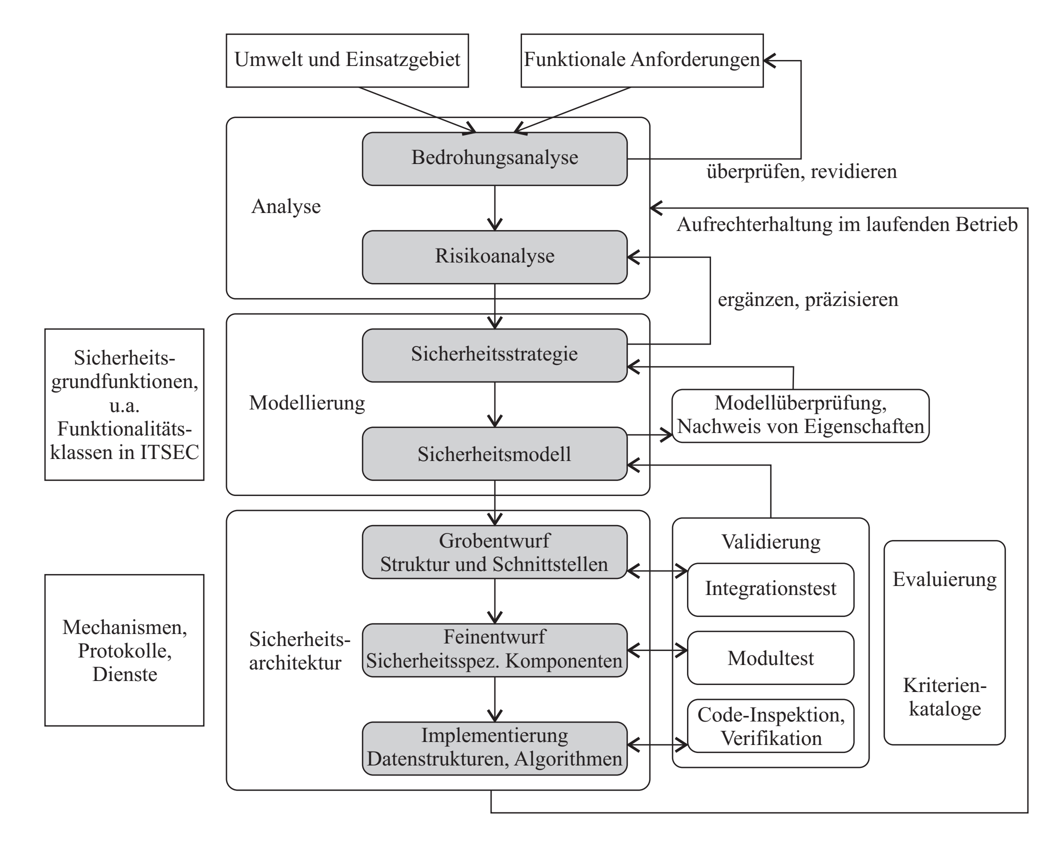


Abbildung 5.1: Phasenstrukturierte Konstruktion sicherer Systeme

Für eine systematische Erfassung der Bedrohungen werden dabei eine Bedrohungsmatrix oder ein Bedrohungsbaum verwendet. Nach der Erfassung der Bedrohungen werden Wahrscheinlichkeiten für das Eintreten innerhalb der Risikoanalyse ermittelt. Der potenzielle Schaden, der verursacht werden kann, wird dabei abgeschätzt.

Aus den Bedrohungen mit ihren Risiken werden die Sicherheitsanforderungen abgeleitet, d.h., Maßnahmen zur Abwehr der Bedrohungen sind zu ermitteln und zu klassifizieren, beispielsweise nach Wichtigkeit, Kosten und Aufwand [Eck03]. In der Sicherheitsstrategie sind die Sicherheitsanforderungen informal oder formal zu beschreiben. Zur Abdeckung der Sicherheitsbedürfnisse können Sicherheitsgrundfunktionen (vgl. Abschnitt 2.2.5) eingesetzt werden, soweit diese vorhanden sind. Weiterhin können hier Funktionsklassen für Anwendungsklassen aus Kriterienkatalogen, wie ITSEC [ITS90] oder Common Criteria (CC) [BSI99], angewandt werden. In solchen Kriterienkatalogen sind für häufig genützte Anwendungsfelder Sicherheitsanforderungen klassifiziert, sodass der Anwender die darin vorgeschlagenen Grundfunktionen zur Abdeckung der Anforderungen übernehmen und mit seinen eigenen Sicherheitsfunktionen kombinieren kann.

Bei der Konstruktion des Sicherheitsmodells werden die funktionalen und sicherheitsbezogenen Eigenschaften auf hohem Niveau modelliert, d.h., von Realisierungseigenschaften, wie beispielsweise von konkreten Protokollen, wird abstrahiert. Auf Basis der Modellfestlegungen wird eine Überprüfung der geforderten Eigenschaften formal oder informal ermöglicht, zum Beispiel können die Vollständigkeit und Konsistenz nachgewiesen werden.

Die zu schützenden Komponenten werden im Grobentwurf identifiziert und die Sicherheitskomponenten sind zu definieren, sodass diese im Feinentwurf erweitert und in der Implementierung die notwendigen Datenstrukturen und Algorithmen gewählt werden können.

Methodisches Testen und Evaluieren sowie eine Verifizierung der sicherheitsrelevanten Funktionen sind in der Validierung durchzuführen, die Tests sind zu dokumentieren und die Vollständigkeit der Abläufe und Testszenarien zu begründen. Das System mit seiner Dokumentation und Einsatzumgebung kann einer Evaluierung durch Dritte unterzogen werden, sodass eine Sicherheitsklassifikation gemäß nationaler oder internationaler Kriterienkataloge erfolgen kann. Dadurch wird Betreibern und Anwendern ein mit der Klassifikationsstufe festgelegtes Maß an Sicherheit zugesichert.

Bewertung

Dieses phasenstrukturierte Vorgehensmodell betrachtet Sicherheitsaspekte durchgehend. Im Gegensatz zu den üblichen Entwicklungspraktiken wird bereits in den frühen Phasen die Sicherheit systematisch erarbeitet. Hierzu sind in dem vorgestellten Prozess explizit Phasen zur Ermittlung von Bedrohungen und Risiken sowie zur Entwicklung einer Sicherheitsstrategie und eines -modells eingeführt worden.

Aspekte der Sicherheit werden über allen Phasen strukturiert erarbeitet, aus den Bedrohungen und Risiken wird systematisch eine Sicherheitsstrategie und ein Sicherheitsmodell erarbeitet, welches in eine Architektur transformiert wird. Abschließende Tests und eine mögliche Evaluierung folgen dieser strukturierten Erarbeitung. Bedrohungsbäume mit Attributierung der Risiken werden zur vollständigen Ermittlung der Bedrohungen und Risiken verwendet. Zur Abwehr der Bedrohungen sind bei der Entwicklung der Sicherheitsstrategie und des Sicherheitsmodells geeignete Maßnahmen zur Abwehr der Bedrohungen zu ermitteln und im

Entwurf zu integrieren. Durch eine derartige strukturierte Entwicklung und Umsetzung der Sicherheitsanforderungen können zugriffssichere Systeme entwickelt werden.

Dieses phasenorientierte Vorgehen trennt jedoch in den frühen Phasen der Entwicklung die funktionalen Aspekte von denen der Sicherheit. Eine gemeinsame Sicherheitsanalyse und Sicherheitsmodellierung mit Entwicklung und Umsetzung der funktionalen Anforderungen ist im Vorgehensmodell nicht vorgesehen, sodass Synergien nicht ausgenutzt werden. Bei der Erarbeitung der Funktionalität können bereits lokale sicherheitsspezifische Aspekte feststellt und herausarbeitet werden. Ebenfalls wäre es möglich, dass bei der Ausarbeitung der Sicherheitsaspekte bereits Sicherheitsgrundfunktionen ermittelt werden und zu den funktionalen Anforderungen hinzugefügt werden. So wäre es beispielsweise möglich, dass bei der Bearbeitung der Sicherheitsstrategie aus Gründen der Verbindlichkeit ein Mechanismus zur Beweissicherung ermittelt und zu den Anforderungen hinzugefügt wird. Datenobjekte und Abläufe für eine Beweissicherung unterliegen ebenfalls Sicherheitsbetrachtungen, die durch eine derartige Verzahnung bereits schrittweise erarbeitet werden können. Wünschenswert wäre hier eine strukturierte Erarbeitung von Aspekten der Sicherheit ähnlich dem Konzept der Anwendungsfälle bei der Entwicklung von funktionalen Anforderungen.

Iterationen sind in dem vorgestellten Vorgehensmodell analog zum Wasserfallmodell nur zwischen der aktuellen Phase und der vorausgehenden Phase vorgesehen. In heutigen objektorientierten Vorgehensmodellen, wie beispielsweise dem Unified Process [JBR99] oder dem Catalysis Approach [DW98], werden Iterationsmöglichkeiten über einzelne oder mehrere Entwicklungsphasen oder über den gesamten Prozess möglich. Die Erweiterung des vorgestellten Ansatzes um übergreifende Iterationsmöglichkeiten ist sinnvoll und notwendig, da durch die Umsetzung von Maßnahmen zur Abwehr von Bedrohungen neue Bedrohungen und Risiken geschaffen werden. Diese können in einer weiteren Iteration behandelt werden. Beispielsweise wäre es sinnvoll, die Analyse- und Modellierungsphase des vorgestellten Ansatzes (vergleiche Abbildung 5.1) iterativ durchzuführen.

Ein weiterer Nachteil des vorgestellten Prozesses ist, dass der vorgestellte Prozess klassisch mit der Entwicklung des Pflichtenheftes beginnt. Umwelt und Einsatzgebiet sowie die funktionalen Anforderungen werden dabei spezifiziert. Bei dieser Art der Ermittlung der Anforderungen werden Sicherheitsaspekte im Pflichtenheft nur sehr oberflächlich ausgedrückt, zum Beispiel dass die Sicherheit als Produktqualität mit "gut" spezifiziert wird (vgl. [Bal96]). Eine frühzeitige Spezifikation von sicherheitsrelevanten Daten und Abläufen, die zumeist bei der Ausarbeitung der funktionalen Anforderungen schon angesprochen werden, findet somit nicht statt. Stattdessen werden diese Informationen nur zu einem Attribut abstrahiert und müssen anschließend erneut erarbeitet werden.

Auf zu entwickelnde und benötigte Produktartefakte geht das phasenstrukturierte Modell nur begrenzt ein. Es werden das Pflichtenheft, die Sicherheitsstrategie und das Sicherheitsmodell genannt, weitere Spezifikationen wie beispielsweise die Dokumentation der Bedrohungen und Risiken sowie eine Umsetzung dieser zu in funktionalen Anforderungen wird nicht beschrieben. Für einen durchgängigen und sukzessiven Entwicklungsprozess sicherer Systeme, in dem in den Prozessschritten Sicherheitsaspekte erarbeitet werden, ist eine Betrachtung der Produktartefakte notwendig.

5.1.2 V-Modell und ITSEC

Ein weiteres Vorgehen zur Entwicklung sicherer Systeme existiert als Erweiterung des V-Modells 97. Hierzu wird das Vorgehen aus dem V-Modell 97 um Aspekte der Entwicklung und Evaluierung sicherer Systeme erweitert. Die in [GDB] beschriebene Anwendung des V-Modells 97 und der ITSEC, genannt SEC, verfolgt das Ziel, IT-Systeme nach dem V-Modell 97 zu entwickeln sowie nach Kriterien des Kriterienkataloges zur Evaluierung sicherer Systeme ITSEC [ITS90] zu bewerten. Da unser Fokus im Rahmen dieser Arbeit auf der Entwicklung von sicheren Systemen, insbesondere in den frühen Phasen der Anforderungsspezifikation, liegt, betrachten wir weniger die Aspekte der Evaluierung und beschränken uns auf die Entwicklung des Systems.

Das V-Modell 97 [DW99, IAB] regelt die Gesamtheit aller Aktivitäten und Produkte sowie die Produktzustände und logischen Abhängigkeiten zwischen Aktivitäten und Produkten im Systementwicklungsprozess sowie die Instandhaltung und Modifikation von existierenden IT-Systemen. Das V-Modell beschreibt dabei den Entwicklungsprozess aus funktionaler Sicht und geht nicht auf spezielle Organisationsformen verschiedenster Einrichtungen bzw. Firmen ein, da es dadurch den Anspruch des universellen Einsatzes gerecht werden will. Weiterhin ist es in 4 Submodelle gegliedert, die sich mit den Aspekten der Systemerstellung (SE), der Qualitätssicherung (QS), des Konfigurationsmanagements (KM) und des Projektmanagements (PM) beschäftigen. Während sich das Submodell QS verstärkt mit der Evaluierung von Systemen beschäftigt, liegt unser Interesse im Submodell SE, da hier die Entwicklung beschrieben wird. Im Folgenden geben wir deshalb einen Abriss der sicherheitsspezifischen Aspekte aus dem Submodell der Systemerstellung.

In der System-Anforderungsanalyse sind nach SEC eine Ist-Aufnahme/Ist-Analyse durchzuführen, das Anwendungssystem zu beschreiben, die Bedrohungen und Risiken zu analysieren sowie das System fachlich zu strukturieren. In einer Ist-Aufnahme-/Ist-Analysephase sind, falls in einem vorhandenen System bereits Sicherheitsmaßnahmen existieren, diese hinsichtlich eines neu vorgegebenen Sicherheitsziels und einer veränderten Einsatzumgebung vorab zu analysieren und zu bewerten. Bei der Beschreibung des Anwendungssystems ist zusätzlich zu den operationellen Anforderungen an das System das Sicherheitsziel vom Auftraggeber vorzugeben. Dieses stellt die Grundlage für die Entwicklung der Sicherheitsanforderungen dar, die in der Phase der Bedrohungs- und Risikoanalyse erarbeitet werden. Dabei werden auf Basis der operationellen Anforderungen und im Hinblick auf das das zu erreichende Sicherheitsziel die Bedrohungen festgestellt und die Risiken ermittelt. Es werden diejenigen Bedrohungen als relevant eingestuft, die ein nicht akzeptables Risiko bergen. Um diese als relevant eingestuften Bedrohungen abzuwehren, müssen geeignete Maßnahmen ergriffen werden. Die Gesamtheit aller Maßnahmen bildet das Sicherheitskonzept. Im Rahmen der fachlichen Strukturierung werden die Maßnahmen durch Sicherheitsfunktionen realisiert, sodass sie den Bedrohungen entgegenwirken können. Diese Funktionen werden als "sicherheitsspezifische Funktionen" bezeichnet, da sie direkt zur Sicherheit des Systems beitragen.

Im Systementwurf ist das System technisch zu entwerfen, die Realisierbarkeit zu untersuchen sowie die Anwenderforderungen zuzuordnen. Beim technischen Systementwurf sind neben den sicherheitsspezifischen Funktionen auch solche Funktionen gegeben, deren korrektes Verhalten für die Sicherheitsfunktionen unerlässlich ist. Diese "sicherheitsrelevanten Funktionen" tragen indirekt zur Sicherheit bei, wenn sich eine oder mehrere Sicherheitsfunktionen auf diese

Funktionen verlassen. Beim Systementwurf ist darauf zu achten, dass sowohl die Sicherheitsfunktionen als auch die sicherheitsrelevanten Funktionen von den nicht als sicherheitsrelevant angesehenen Funktionen ausreichend abgegrenzt werden, sodass der Umfang des gesamten Sicherheitsanteils so klein wie möglich gehalten werden kann. In der Realisierbarkeitsuntersuchung ist die Abgrenzbarkeit des sicherheitsrelevanten Teils vom nicht sicherheitsrelevanten zu überprüfen. Die Machbarkeit der geforderten Sicherheitsstufe nach ITSEC ist zu untersuchen, wobei die erwartete Komplexität des Sicherheitsanteils ein Kriterium für die Realisierbarkeit darstellt. Es sind Überlegungen bezüglich der Wirtschaftlichkeit anzustellen. Die Untersuchungen, die durchzuführen sind, müssen unter Berücksichtigung des tragbaren Restrisikos die Kosten und die möglichen Schäden, die beim Wirksamwerden der Bedrohungen eintreten, abwägen. Beim Zuordnen der Anwenderforderungen ist der Nachweis zu erbringen, dass sämtliche Sicherheitsanforderungen im Sicherheitskonzept berücksichtigt werden. Dadurch kann gegebenenfalls ein vorhandenes Restrisiko festgestellt werden.

In der Software-/Hardware-Anforderungsanalyse ist zu überprüfen, inwiefern sich sicherheitsrelevante Teile auf die Software-/Hardware-Einheiten auswirken. Falls sicherheitsspezifische oder sicherheitsrelevante Funktionen in der Software-/Hardware-Einheit realisiert werden müssen, muss innerhalb der Definition von allgemeinen Anforderungen aus Sicht der SW-/HW-Einheit eine exakte Schnittstellenbeschreibung zum restlichen sicherheitsspezifischen oder sicherheitsrelevanten Teil, der in anderen SW-/HW-Einheiten verstreut sein kann, erstellt werden. Innerhalb der Definition der Anforderungen an die Funktionalität sind die Anforderungen an die Sicherheitsfunktionen und die dafür notwendigen Daten zu erstellen. Weiterhin sind die Anforderungen an die Qualität der SW-/HW-Einheit abhängig von der Kritikalitätsstufe zu erarbeiten.

Die Softwarearchitektur ist innerhalb des Softwaregrobentwurfs zu entwerfen. Dabei ist die Bedrohungs- und Risikoanalyse fortzuschreiben und zu untersuchen, inwiefern durch die Verfeinerung neue Bedrohungen und Risiken auftreten. Abhängig von der gewählten Sicherheitsstufe nach ITSEC wird eine textuelle, semiformale oder formale Architekturbeschreibung der sicherheitsrelevanten Teile gefordert. Weiterhin sind im Grobentwurf die internen und externen Softwareschnittstellen zu entwerfen und die Softwareintegration zu spezifizieren. Es sind alle Schnittstellen der sicherheitsspezifischen und sicherheitsrelevanten SW-Komponenten, SW-Prozesse und SW-Module mit ihrem Zweck und ihren Parametern zu beschreiben. Dabei ist die Separierung des nicht sicherheitsrelevanten Teils zu verdeutlichen und die Schnittstellen sind zu minimieren.

Im Feinentwurf sind die Komponenten, Module und die Datenbank der Software abhängig von der gewählten Kritikalitätsstufe informal oder semiformal zu beschreiben und für Sicherheitsvorgaben sind der Betriebsmittel- und Zeitbedarf gegebenenfalls zu ermitteln.

Innerhalb der Selbstprüfung der SW-Module und der Datenbank ist ein Abgleich des Codes mit dem Feinentwurf durchzuführen und in der Systemintegration ist das Zusammenwirken der Mechanismen zu untersuchen. In der Produktbereitstellung muss die korrekte Anwendung der Sicherheitsfunktionen durch den Sicherheitsverantwortlichen und die Benutzer in Handbüchern geeignet dokumentiert werden.

Abbildung 5.2 gibt einen Überblick über die Aktivitäten, die bei der Entwicklung von sicheren Systemen und deren Evaluierung nach SEC auszuführen sind. Die grau hinterlegten Rechtecke markieren die Aktivitäten, die zur Systementwicklung notwendig sind, die weiß

SE1 System-Anforderungsanalyse	Ist-Aufnahme/-Analyse durchführen	Anwendungssystem beschreiben	System fachlich strukturieren	Bedrohungen und Risiken analysieren	
SE2 Systementwurf	System technisch entwerfen	Realisierbarkeit untersuchen	Anforderungen zuordnen	Systemintegration spezifizieren	
SE3 SW/HW-Anforderungsanalyse	Allg. Anforderungen aus Sicht der SW-/HW-Einheit definieren	Anf. an die externen Schnittstellen der SW-/HW-Einheit präzisieren	Anforderungen an die Funktionalität definieren	Anforderungen and die Qualität der SW-/HW-Einheit definieren	Anforderungen an die Entwicklungsumgebung definieren
SE4 SW-Grobentwurf	SW-Architektur entwerfen	SW-interne und -externe Schnittstellen entwerfen	SW-Integration spezifizieren		
SE5 SW-Feinentwurf	SW-Komponente, SW-Modul, Datenbank beschreiben	Betriebsmittel- und Zeitbedarf analysieren			
SE6 SW-Implementierung	Selbstprüfung des SW-Moduls/der Datenbank durchführen				
SE8 Systemintegration	Selbstprüfung des Systems durchführen	Produkt bereitstellen			
PM1 Projektonitialisierung	Projektkriterien und Entwicklungsstrategie festlegen	Projektspezifisches V-Modell erstellen	Toolset-Management durchführen	Grobplan erstellen	Schulung und Einarbeitung
QS1 QS-Initialisierung			**KM1 KM-Planung**		
QS2 Prüfungsvorbereitung	Prüfungsmethoden und -kriterien festlegen		**KM2 Produkt- und Konfigurationsverwaltung**		
QS3 Produktprüfung	Produkt inhaltlich prüfen		**KM 3 Änderung-management**	Änderung abschließen	

Abbildung 5.2: Aktivitäten bei Anwendung des V-Modells und der ITSEC

hinterlegten beschäftigen sich mit weiteren Aufgaben, die in Zusammenhang mit der Evaluierung (Qualitätssicherung, Konfigurationsmanagement) oder mit dem Projektmanagement stehen.

Bewertung

Die Sicherheitseigenschaften werden in SEC durchgehend über die gesamte Systementwicklung und Evaluation betrachtet. In allen aufgezeigten Entwicklungsphasen, von der Anforderungsanalyse bis hin zur Systemintegration, werden Sicherheitsaspekte erarbeitet und eingearbeitet. Beginnend von einer Bedrohungs- und Risikoanalyse wird ein Sicherheitskonzept entwickelt und Sicherheitsfunktionen sowie sicherheitsrelevante Funktionen werden darauf aufbauend im System- und im Softwareentwurf zu sicherheitsrelevanten SW-Komponenten, -Prozessen und -Moduln erarbeitet, wobei eine strikte Abgrenzung des sicherheitsrelevanten Teils von dem allgemeinen Teil gefordert wird. Dadurch, dass die in einer Phase erarbeiteten Sicherheitsaspekte in der nächsten Phase wieder verwendet und zur Entwicklung neuer Sicherheitsaspekte eingesetzt werden, findet eine sukzessive Entwicklung der Sicherheitsaspekte statt, in der in den verschiedenen Phasen die Sicherheit nicht "von der grünen Wiese" spezifiziert wird, sondern die bereits erarbeiteten Ergebnisse strukturell umgesetzt werden.

Funktionalität und Sicherheit werden in diesem Modell gemeinsam entwickelt, denn innerhalb des Prozesses werden aufeinander aufbauend sowohl operationelle als auch sicherheitsgetriebene Aspekte betrachtet. Durch diese Arbeitsweise werden Ergebnisse der operationellen Analyse der Sicherheitsmodellierung zugeführt und die daraus entstehenden Resultate in der operationalen Systemmodellierung mit einbezogen. Im Gegensatz zu einer strikt getrennten Vorgehensweise können somit frühzeitige gegenseitige Beeinflussungen (engl. *feature interaction*) erkannt und modelliert werden. Ein Beispiel hierfür ist die Vorgabe der operationalen Anforderungen sowie des Sicherheitsziels und die darauf folgende fachliche Strukturierung mit Umsetzung der Maßnahmen aus dem Sicherheitskonzept.

Die Sicherheitsaspekte werden einerseits verschränkt mit der Funktionalität entwickelt, andererseits wird die Sicherheit nicht lokal bei der Bearbeitung der operationalen Aspekte spezifiziert. Auf Aktivitäten der funktionalen Modellierung folgen Aktivitäten zur Sicherheitsmodellierung und umgekehrt. Aber innerhalb kleiner Bearbeitungsschritte, wie beispielsweise der Ausarbeitung einer kleinen, funktionalen Einheit, werden hierzu nicht sofort die lokal auftretenden Sicherheitseigenschaften analysiert. Sicherheitsaspekte werden bei der Bearbeitung von funktionalen Einheiten spezifiziert. Nach diesem Ansatz werden jedoch sicherheitsrelevante Erkenntnisse, die sich bei der detaillierten funktionalen Modellierung ergeben, vorerst "über Bord geworfen" und müssen anschießend nochmals erarbeitet werden. Wünschenswert wäre hier, dass die Erkenntnisse bereits vorab spezifiziert werden und in einem nachfolgenden Schritt bei der Bearbeitung von größeren funktionalen Einheiten nochmals bezüglich Konsistenz und Vollständigkeit überprüft bzw. nachgebessert werden.

Im Vordergrund bei der Beschreibung von SEC stehen die Aktivitäten, die zur Sicherheitsmodellierung durchgeführt werden. Produktartefakte, die während der Sicherheitsmodellierung anzufertigen sind, werden nicht angegeben, sodass eine durchgängige Entwicklung von Artefakten nicht gegeben ist. Es werden zum Beispiel die Bedrohungen und Risiken innerhalb der Anforderungsanalyse ermittelt, jedoch wird kein Artefakt angegeben, in dem diese Sicherheitsaspekte dokumentiert werden. Einzig und allein kommt ein Sicherheitskonzept als Gesamtheit aller Maßnahmen innerhalb der Aktivitätsbeschreibungen vor.

Iterative Entwicklung wird in SEC nur an einer einzigen Stelle erwähnt. Beim Entwurf der Softwarearchitektur innerhalb des Softwaregrobentwurfs ist zu untersuchen, inwiefern durch die Verfeinerung weitere Bedrohungen und Risiken entstehen. Neben dieser und den generellen Iterationsmöglichkeiten des V-Modells 97 wird nicht angegeben, wie die Sicherheit in den Phasen der Software-Anforderungsanalyse, des Systementwurfs, der SW-/HW-Anforderungsanalyse sowie in Feinentwurf, Implementierung und Integration umgesetzt werden kann.

Auf eine Sicherheitsmodellierung in der Geschäftsprozessanalyse geht SEC ebenfalls nicht ein. SEC startet mit der Systemanforderungsanalyse und benötigt hierzu die operationellen Anforderungen an das System und das Sicherheitsziel vom Auftraggeber.

5.1.3 Ein Prozess konform zu den Common Criteria

Ein weiteres Vorgehen zur Entwicklung sicherer Systeme wurde innerhalb einer Diplomarbeit zum Thema *Security Engineering nach den Common Criteria – Eine Fallstudie* (CC-Vorgehen) [Vet01, VWW02] ausgearbeitet, welches innerhalb des Softwaretechnikpraktikums

PalME im Sommersemester 2001 an der TU München exemplarisch angewendet wurde. Innerhalb *PalME* wurde eine sichere elektronische Geldbörse für den Palm Pilot mit einer Gruppe von 18 Studenten nach dem in der Diplomarbeit vorgestellten Security Engineering Prozess entwickelt. Das Vorgehen war auf eine anschließende Sicherheitsevaluierung nach den Common Criteria [BSI99], Sicherheitsstufe EAL2 ausgerichtet. Es wurden im Entwicklungsprozess alle notwendigen Spezifikationen für eine Sicherheitsevaluierung in die Entwicklung mit einbezogen, die nach der Sicherheitsstufe 2 der siebenstufigen Gliederung gefordert werden, und diese Dokumente wurden im weiteren Entwicklungsprozess verwendet. Durch dieses Vorgehen setzen sich die Entwickler frühzeitig mit Sicherheitsaspekten auseinander und können so früher auf Fehler oder Sicherheitslücken reagieren und Gegenmaßnahmen einleiten. Eine Erkennung von Fehlern und Sicherheitslücken in späten Phasen des Entwicklungsprozesses ist immer mit hohem Zeit- und Kostenaufwand verbunden und kann somit vermieden werden.

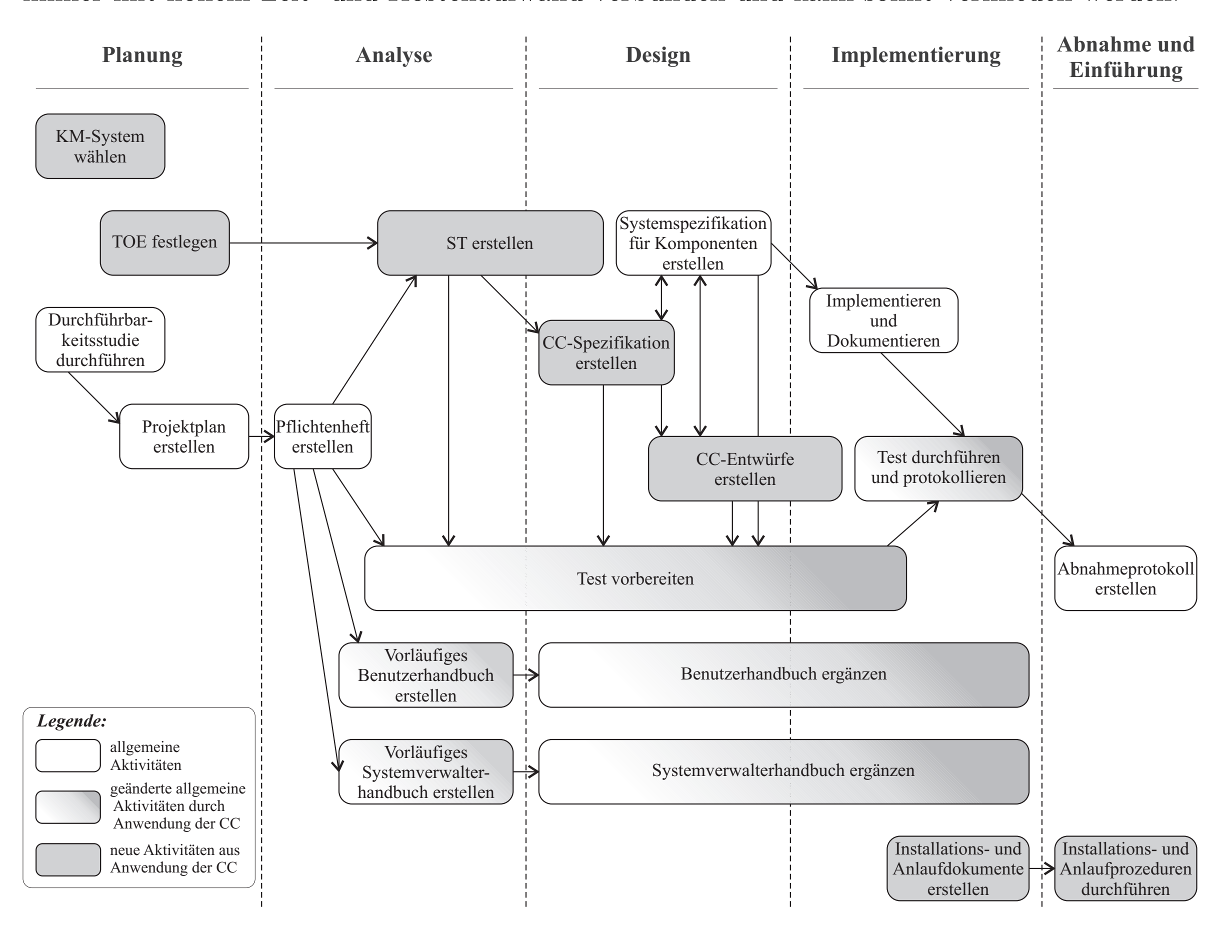


Abbildung 5.3: Ein zu den Common Criteria konformes Vorgehen

Als Grundlage für das CC-Vorgehen wurde das klassische Wasserfallmodell [Roy70] ausgewählt, da es einfach strukturiert, für kleine Anwendungen geeignet ist und in Projekten angewendet werden kann, in denen die Anforderungen zu Beginn der Entwicklung bekannt sind. Dieses phasenorientierte Vorgehensmodell wurde mit den sicherheitsspezifischen Aktivitäten angereichert, die zur Entwicklung eines sicheren Systems mit anschließender Evaluierung nach den Common Criteria notwendig sind. Abbildung 5.3 zeigt das um Sicherheitsaspekte abgeänderte und ergänzte, auf dem Wasserfallmodell basierte CC-Vorgehen. Dabei stellen

die weiß hinterlegten Aktivitäten diejenigen Arbeitsschritte dar, die in einem herkömmlichen Projekt ohne Sicherheitsbetrachtungen nach dem Wasserfallmodell zu bearbeiten sind. Solche Arbeitsschritte, die aufgrund der Sicherheitsbetrachtungen zusätzlich in das Vorgehensmodell integriert wurden, sind in der Abbildung grau hinterlegt. Zudem beeinflusst die Betrachtung der Sicherheitsaspekte auch einige herkömmliche Aktivitäten des Wasserfallmodells dahingehend, dass sie zusätzlich Anforderungen der CC abdecken müssen und deshalb in ihrer gewohnten Ausführung geändert werden mussten. Diejenigen Aktivitäten des Wasserfallmodells, die bezüglich der CC geändert werden mussten, sind weißgrau hinterlegt.

Zu Beginn der Entwicklung muss zunächst der Evaluierungsgegenstand (engl. *Target of Evaluation, TOE*) festgelegt werden, der zusätzlich zu den allgemeinen funktionellen Anforderungen die Sicherheitsanforderungen erfüllen muss. Ein TOE kann sowohl das gesamte zu entwickelnde System umfassen oder aber nur einen Teil davon als zu evaluierenden Systemteil beinhalten. Weiterhin ist in der Planungsphase nach dem CC-Vorgehen neben der Durchführbarkeit und Aufwandsschätzung auch der Einsatz eines Konfigurationsmanagementsystems (engl. *Configuration Management System, CMS*) festzulegen, sodass eine eindeutige Identifikation des Evaluierungsgegenstands möglich ist.

Die Analysephase beinhaltet die Formulierung der fachlichen Anforderungen in Form einer Systemanalyse, die Erstellung eines vorläufigen Benutzerhandbuchs und eines Produktmodells, die Erstellung der Sicherheitsvorgaben (engl. *Security Target, ST*) sowie die Anfänge der Testvorbereitung. In den Sicherheitsvorgaben sind die Bedrohungen, Annahmen und Sicherheitspolitiken des TOE auszuwählen, Sicherheitszeile und Sicherheitsfunktionen zu formulieren und die funktionalen Sicherheitsanforderungen zu selektieren. Zudem muss bereits in dieser Phase eine vorläufige Version des Benutzerhandbuchs und des Systemverwalterhandbuchs erstellt werden, in denen die bereits bekannten Anforderungen zum TOE beschrieben und später in der Designphase ergänzt werden. Da Testabläufe teilweise schon während der Analyse bekannt sind, insbesondere die Testabläufe aus der Sicht des Benutzers, werden diese bereits in dieser frühen Phase in den Testvorbereitungen involviert.

In der Designphase werden, neben der Systemspezifikation für die Komponenten, eine CC-Spezifikation und ein CC-Entwurf gefordert. Ausgehend von den Sicherheitsvorgaben werden die Sicherheitsfunktionen auf den Ebenen des Grob- und Feinentwurfs spezifiziert und Entwürfe erstellt. Wie fein granular die Spezifikationen und Entwürfe auszuführen sind, hängt von der gewählten Evaluierungsstufe ab, die zu Projektbeginn festgesetzt werden muss. Ebenso sind in der Designphase auch das Benutzer- und Systemverwalterhandbuch fortzuschreiben.

Neben der eigentlichen Implementierung werden in der Implementierungsphase Tests ausgeführt sowie Installations- und Anlaufdokumente erstellt. In den Tests muss nachgewiesen werden, dass der Evaluierungsgegenstand die geforderten funktionalen Sicherheitsanforderungen tatsächlich erfüllt. Hierzu ist zu zeigen, dass sich die Sicherheitsfunktionen des TOE gemäß ihrer Spezifikationen verhalten. Die Evaluierungsstufe legt den Testumfang, die Testtiefe sowie die Abdeckung der funktionalen Anforderungen beim Testen fest. Das Benutzer- und Systemverwalterhandbuch ist fertig zu stellen, der Installationsvorgang ist zu beschreiben und die notwendigen Anlaufdokumente müssen bereitgestellt werden, sodass die Installation in der Abnahme- und Einführungsphase durchgeführt und das Abnahmeprotokoll des Systems erstellt werden kann.

Bewertung

Wie auch in den beiden bereits vorgestellten Vorgehensmodellen zur Entwicklung von Sicherheitsaspekten betrachtet das CC-Vorgehen die Sicherheitsaspekte durchgehend im Prozess. Ab dem Beginn des CC-Vorgehens in der Planungsphase bis hin zur Implementierung mit Abnahme und Einführung werden in allen Phasen Arbeitsschritte zur Sicherheitsmodellierung vorgestellt, sodass in keiner Phase Sicherheitsaspekte unbetrachtet bleiben. Bei dieser durchgängigen Betrachtungsweise wird die Sicherheit ebenfalls wie im phasenstrukturierten und im V-Modell-konformen Vorgehen sukzessive erarbeitet, d.h., die in Phase n erarbeiteten Sicherheitsaspekte werden in der $(n + 1)$ten Phase wieder aufgegriffen und weiter verfeinert bzw. zur Erarbeitung der Funktionalität herangezogen. So fließen zum Beispiel die Sicherheitsvorgaben aus der Analysephase in die CC-Spezifikation und in die Testvorbereitung während der Designphase mit ein und die CC-Entwürfe folgen der Ausarbeitung der CC-Spezifikation.

Die Entwicklung der Sicherheitsaspekte ist in die funktionale Entwicklung integriert, sodass gegenseitige Einwirkungen betrachtet werden. Die Ergebnisse der funktionalen Betrachtung fließen in die Sicherheitsanalyse ein und umgekehrt. Beispielsweise fließen Informationen der Komponentenspezifikation in die CC-Spezifikation mit ein. Die CC-Spezifikation bezieht sich auch auf die ausgearbeiteten Komponenten, sodass zum Beispiel erkannt werden kann, dass für eine Teilkomponente Abrechnung eines Warenwirtschaftssystems besondere Bedrohungen zu betrachten sind.

Sicherheitsaspekte werden jedoch im CC-Vorgehen immer als Ganzes betrachtet. Im Rahmen der Bearbeitung der Sicherheitsvorgaben sowie bei der CC-Spezifikation und -Evaluierung wird immer die Gesamtheit aller Sicherheitsaspekte betrachtet. Eine Modellierung eines Ausschnitts der Sicherheit mit anschließender Integration in das Gesamtsystem, die den Vorteil der Überschaubarkeit und der frühzeitigen Reaktionsmöglichkeit mit sich bringen würde, findet in der vorgestellten Vorgehensweise nicht statt.

In diesem Vorgehensmodell sind neben den Prozessaktivitäten auch Produktartefakte angegeben, die während des Sicherheitsprozesses zu entwickeln sind, so etwa die Sicherheitsvorgaben, die CC-Spezifikation oder die Benutzer- und Systemverwalterhandbücher. Die CC fordern beispielsweise für eine Evaluierung nach Stufe EAL2 neben Beschreibungen über die funktionale Spezifikation und des Entwurfs auf hoher Ebene Nachweise über die Testabdeckung. Jedoch ist die geforderte Menge an Produktartefakten im CC-Vorgehen zu grob granular und unvollständig. So wird zum Beispiel im Design der funktionalen Bestandteile nur eine Systemspezifikation für Komponenten gefordert, die eine Grundlage für die Implementierung darstellen soll. Es fehlen hier beispielsweise noch ein Domänenmodell und eine fein granularere Betrachtung der Sicherheitsartefakte, beispielsweise in einem Bedrohungs- und Risikomodell. Die Produktartefakte müssten so fein granular angegeben sein, dass zu jeder Aktivität und Phase die für die Bearbeitung notwendigen und erzeugten bzw. veränderten Dokumente angegeben werden können.

Auf iterative Aspekte innerhalb einer Ausführungsphase wird im CC-Vorgehen nicht eingegangen. Es wird nur erwähnt, dass für eine iterative Entwicklung der gesamte Prozess mehrfach durchlaufen werden kann. Eine frühzeitige Reaktion auf fehlende oder falsche Anforderungen sowie auf iterative Entwicklung der Sicherheitsaspekte ist in diesem Ansatz nicht vorhanden.

Wie auch bei dem phasenstrukturierten Vorgehen und beim V-Modell/ITSEC-Vorgehen beginnt die Modellierung mit der Planungsphase. Eine Geschäftsprozessmodellierung ist nicht

Gegenstand des CC-Vorgehensmodells, sodass hierzu auch keine weiteren Aussagen getroffen werden können.

5.1.4 Weitere Vorgehensbeschreibungen

Nach dieser ausführlichen Vorstellung und Bewertung von drei Ansätzen zur Entwicklung von sicheren Systemen stellen wir im Folgenden noch drei weitere Ansätze vor, die sich noch in einem frühen Stadium ihrer Entwicklung befinden oder nur ergänzend in Büchern zur Entwicklung sicherer Systeme (vgl. [And01, VM02]) angegeben sind, jedoch einige interessante Gedanken für eine sichere Systementwicklung beinhalten.

Security Engineering in frühen Phasen

Grundprinzipien und Ideen zu einem Security Engineering Process stellt [AW03b] vor. Dabei wird insbesondere das Problem der späten Integration von Sicherheitsaspekten in bestehenden Ansätzen angemahnt und ein Vorgehen zur Modellierung von Sicherheitseigenschaften ab den frühen Phasen umrissen. Ein weiteres Ziel dieses Vorgehens ist es, eine Menge an Richtlinien und Funktionen zur Sicherheit bereitzustellen, sodass gewöhnliche Entwickler, also keine "Sicherheitsexperten", sichere Software erstellen können. Dieser Ansatz ist nicht auf die Entwicklung einer eigenen, vollständigen Methode ausgerichtet, sondern schlägt vor, ein bestehendes objektorientiertes Vorgehensmodell mit Sicherheitsaspekten zu erweitern, besonders in der Phase der Anforderungsspezifikation.

Dieser Ansatz zur Sicherheitsmodellierung beginnt mit der Erarbeitung von Sicherheitsaspekten in der Problembeschreibung, die entweder als textuelle Beschreibung vom Auftraggeber geliefert oder in gemeinsamen Diskussionen von Entwicklern und Anwendern erarbeitet wird. Diese Beschreibung wird in einem ersten Schritt um Sicherheitsanforderungen ergänzt. In [AW03b] wird hierzu eine Matrix vorgestellt, die zu bekannten Sicherheitsanforderungen Sicherheitsmaßnahmen vorstellt. Die so ermittelten Maßnahmen fließen in die Anforderungsspezifikation mit ein. In der Anwendungsfallmodellierung werden die Akteure bestimmt, wobei auch Security-Patterns als Akteure betrachtet und Zugriffsrechte in Form von Vorbedingungen neben der üblichen Anwendungsfallspezifikation beschrieben werden. In Anwendungsfalldiagrammen werden die Sicherheitsmaßnahmen als *UML Notes* dargestellt, in Klassendiagrammen als *UML Stereotypes*, wobei die Stereotypes in *UML Notes* eingetragen sind. In der Analyse wird zwischen drei verschiedenen Klassenkategorien zur Sicherheit, *Entity*, *Boundary* und *Control*, unterschieden und es werden Security Patterns zur Integration der Sicherheitsmaßnahmen angewandt.

In diesen Ansatz wird Sicherheit in allen Projektphasen, beginnend bei der Anforderungsanalyse, durchgeführt und die Sicherheit wird schrittweise im System integriert. Sicherheit wird gemeinsam mit der Funktionalität entwickelt. In den Anwendungsfällen wird die Sicherheit lokal bei der betrachteten Funktionalität spezifiziert. Für die Ausarbeitung der Sicherheit wird kein iteratives Vorgehen explizit angegeben, vielmehr wird auf das iterative Vorgehen des übergeordneten Vorgehensmodells verwiesen und dies auch gefordert. Eine Geschäftsprozessanalyse ist nicht Bestandteil dieses Ansatzes.

Sicherheitsrisiken im Spiralmodell

Zur Behandlung von Sicherheitsrisiken im Softwarelebenszyklus wird in [VM02] eine Anpassung des Spiralmodells [Boe86] vorgestellt. Das Spiralmodell eignet sich zum Management

von Sicherheitsrisiken nach der Auffassung von Viega und McGraw deshalb sehr gut, da die gleichen Entwicklungsaufgaben wieder und wieder durchgeführt werden und somit alle Ergänzungen zur Gewährleistung der Sicherheit einer Sicherheitsanalyse unterzogen werden können. Am Ende entsteht ein vollständiges und robustes Produkt. Besondere Bedeutung liegt in diesem Ansatz auf einer frühzeitigen Betrachtung von Sicherheit, denn bereits bei der Ermittlung der Anforderungen sind Sicherheitsaspekte zu betrachten. Bei der Ausführung der Risikoanalyse und der Erarbeitung von Strategien sind die in den Anforderungen festgelegten Sicherheitsaspekte für eine vollständige Sicherheitsanalyse einzubeziehen. In diesem Ansatz werden in jeder Iteration alle vorliegenden Produktartefakte verwendet und gegebenenfalls erweitert. Dabei sind beispielsweise die Anforderungsbeschreibung, die Architektur- und Designbeschreibung, existierender Code sowie Testfälle wichtige Produktartefakte, die beim Risikomanagement und bei der Umsetzung der Sicherheit anzuwenden sind.

Die Besonderheit bei der Anwendung des Spiralmodells liegt darin, dass in jeder lokalen Iteration eine Folge von Entwicklungsschritten durchgeführt wird, in der die Bedrohungen und Risiken verfeinert werden. Dabei sind die Sicherheitsaspekte nicht in ihrer Gesamtheit zu erarbeiten, sondern ebenfalls lokal zu betrachten, sodass auf alle neuen Probleme, die während der Bearbeitung entstehen können, eingegangen werden kann. Im Gegensatz zum Wasserfallmodell [Roy70] ist dieses Modell bei der Erarbeitung von Sicherheitsrisiken und Maßnahmen flexibler, denn die in frühen Phasen erarbeiteten Ergebnisse sind in die Entwicklung einbezogen, neue Probleme können in die Bearbeitung aufgenommen und gegebenenfalls kann eine Iteration erneut ausgeführt werden, falls am Ende einer Iteration Inkonsistenzen oder Sicherheitslücken festgestellt werden.

Dieser Ansatz erlaubt die durchgängige Betrachtung von Sicherheitsaspekten, denn bereits ab der Anforderungsanalyse werden Sicherheitsaspekte ermittelt und Lösungen entwickelt. Durch das iterative Vorgehen wird die Sicherheit sukzessive entwickelt. Lokale Probleme sind sowohl nach funktionalen Aspekten als auch nach Aspekten der Sicherheit zu betrachten und zu beheben. Der Ansatz zum Spiralmodell betrachtet zudem Produktartefakte im Vorgehensmodell. Leider stellt [VM02] nur knapp die Ideen zur Erarbeitung von Sicherheitsrisiken vor, eine ausführliche Darstellung des Prozesses findet nicht statt. Ebenso fehlten Erfahrungsberichte aus der Praxis, die die Anwendung dieses Vorgehens untermauern.

Security Engineering Management

Ross Anderson beschreibt in seinem Buch mit dem Titel "Security Engineering" [And01] Grundlagen für die Entwicklung von zugriffssicheren Systemen, wie beispielsweise Protokolle, Kryptografie, Fälschungssicherheit, Abhörmöglichkeiten. In einem Abschnitt wendet er sich dem Vorgehen für eine sichere Entwicklung zu und skizziert eine *Top-Down-* und eine *iterative* Vorgehensweise. Bei der ersten Vorgehensweise wird die Software zumeist nach dem Wasserfallmodell entwickelt. Zur Entwicklung der Sicherheit werden dabei die Gefahren des Systems identifiziert, Risiken bewertet und eine Sicherheitsstrategie entwickelt. Die Sicherheitsstrategie wird anschließend auf die Soft- und Hardware übertragen und abschließend finden Tests statt. Nach Anderson eignet sich diese Methode nicht zur Entwicklung von sicheren Systemen, da nicht alle potenziellen Sicherheitslücken gefunden werden.

Zur Entwicklung von zugriffssicheren Systemen plädiert Anderson für eine Problemanalyse im Prozess, in der alle potenziellen Gefahren systematisch ermittelt werden. Mögliche Realisierungen für eine Problemanalyse sind eine top-down orientierte *Analyse mit Bedrohungsbäu-*

men oder eine bottom-up gerichtete *Failure Modes and Effects Analysis*. Die optimale Analyse stellt eine Kombination aus beiden Ansätzen dar.

Für das *Requirements Engineering* führt Anderson Grundsätze an, die bei der Entwicklung zu betrachten sind:

- Für diese Aktivitäten sind Entwickler mit speziellen Fähigkeiten notwendig oder es müssen Experten hinzugezogen werden.
- Sicherheitsaspekte sind innerhalb der funktionalen Anforderungen und der Testdokumentation zu betrachten, denn eine externe Betrachtung der Sicherheit in allein stehenden und isolierten Dokumenten führt dazu, dass diese Dokumente während der Systementwicklung vernachlässigt oder beiseite gelegt werden. Dies führt letztendlich dazu, dass Sicherheitsaspekte nicht weiter verfolgt werden.
- Sicherheitsaspekte müssen in die gesamte Systementwicklung integriert werden. Eine nachträgliche Entwicklung der Zugriffssicherheit funktioniert nicht und führt nicht zur Erfüllung der Sicherheitsanforderungen.
- Bedrohungen sind auch nach Abschluss einer Bedrohungsanalyse weiter zu betrachten, gegebenenfalls in weiteren Iterationen. Erfahrungen haben gezeigt, dass Anforderungen zum Schutz des Systems nicht in einer einzigen Phase richtig und vollständig analysiert werden können.

Nach [And01] ist in der Anforderungsanalyse eine iterative Vorgehensweise nach dem Spiralmodell [Boe86] sinnvoll, da hierbei die Anforderungen an die Sicherheit schrittweise verfeinert werden können. In einer ersten Phase ist das neue System zu beschreiben und gegenüber existierenden Systemen abzugrenzen. Ein Risikomodell und der Entwurf einer Security-Policy dokumentieren die Sicherheitsbetrachtungen. Die Ergebnisse aus der ersten Phase werden dem Kunden zur Überarbeitung vorgelegt und es findet eine interne Bewertung statt. Die Entwickler erweitern ihr Fachwissen durch internes Material und Literaturstudien. Als Ergebnis liefert die zweite Phase überarbeitete, qualitativ hochwertige Risikomodelle, Security-Policies und Security Targets. In der dritten Phase erfolgt eine Verteilung der Dokumente an einen größeren Kreis, wie etwa an Senior-Manager, externe Bewerter oder Evaluatoren.

Im Rahmen der Entwicklung zugriffssicherer Systeme sind nach Anderson ein Bedrohungsmodell mit Angreifern und Schwachstellen sowie ein Strategiemodell zu erstellen. Das Strategiemodell wird in die Beschreibung des *Sicherheits-Targets* (vgl. [BSI99]) aufgenommen, welches Basis für die weitere Entwicklung sowie das Testen und gegebenenfalls für eine Evaluierung ist.

Die von Ross Anderson vorgestellten Aspekte sind nur Teile eines Vorgehens für eine Entwicklung sicherer Systeme, denn nur im Bereich des Requirements Engineerings geht er ansatzweise auf Einzelheiten ein. Ein Vorgehen in anderen Phasen, wie beispielsweise in der Analyse oder im Design fehlt vollständig. Die im Folgenden beschriebene Bewertung kann deshalb nur die betrachteten Aspekte umfassen.

Im Requirements Engineering erfolgt eine durchgängige und sukzessive Entwicklung der Sicherheitseigenschaften, denn aus den Bedrohungen und Risiken werden eine Strategie sowie die Umsetzung in ein Sicherheits-Target abgeleitet. Für eine gemeinsame Entwicklung von

Funktion und Sicherheit plädiert Anderson, jedoch fehlt es an detaillierten Arbeitsschritten. Produktartefakte und Iterationsmöglichkeiten werden im Rahmen des Requirements Engineerings diskutiert.

5.2 Prozessanforderungen an die Entwicklung zugriffssicherer Systeme

Nachdem wir in den vorausgegangenen Abschnitten existierende Vorgehensmodelle zur Entwicklung zugriffssicherer Systeme bezüglich ihrer Stärken und Schwächen analysiert haben, stellen wir im Folgenden Prozessanforderungen zur Entwicklung zugriffssicherer Systeme vor und fassen im Abschnitt 5.2.2 tabellarisch die Anforderungsabdeckung für die im Abschnitt 5.1 vorgestellten Vorgehensmodelle zusammen.

5.2.1 Prozessanforderungen

Bei den im Folgenden aufgelisteten Prozessanforderungen zur Entwicklung zugriffssicherer Systeme handelt sich um solche, die größtenteils für den gesamten Prozess gültig sind. Da wir uns in dieser Arbeit jedoch besonders für die Phasen der Anforderungsspezifikation interessieren, heben wir primär die Prozessanforderungen an die frühen Phasen zur Entwicklung zugriffssicherer Systeme hervor.

Die Prozessanforderungen zur Entwicklung zugriffssicherer Systeme werden durchnummeriert (SPAx), sodass sie später in der Arbeit referenziert werden können.

SPA1: Durchgängige Sicherheitsbetrachtungen

Derzeitige Entwicklungen von zugriffssicheren Systemen scheitern oftmals daran, dass Sicherheit erst in späten Phasen der Entwicklung betrachtet wird. Die Systementwicklung erfolgt vorab größtenteils ohne Analyse von Sicherheitsaspekten und im Nachhinein wird versucht, diese zu integrieren. Dies führt entweder zu einem zusätzlichen Arbeitsaufwand, der dadurch entsteht, dass bei der Integration der Sicherheitsaspekte in den späteren Phasen Aktivitäten aus den früheren Phasen wiederholt werden müssen, oder dazu, dass Aspekte der Zugriffssicherheit nicht in dem erforderlichen Maß betrachtet werden. Durch eine durchgängige Entwicklung von Aspekten der Zugriffssicherheit kann die Sicherheitsanalyse auf Änderungen und Erweiterungen eingehen und Sicherheitslücken können vermieden werden. Ungeeignete Maßnahmen zur Gewährleistung der Sicherheit können erkannt werden, in dem beispielsweise in der Phase des Testens die gewählten Maßnahmen zur Abdeckung der Zugriffssicherheit bezüglich ihrer Tauglichkeit überprüft werden.

SPA2: Sukzessive Erarbeitung von Sicherheit

So wie es bei einer Softwareentwicklung entscheidend zum Erfolg beiträgt, dass die ausgearbeitete Spezifikation als Grundlage zum Design und zur Implementierung verwendet wird, ist es bei der Entwicklung von Aspekten der Zugriffssicherheit ebenfalls wichtig, die erstellten Spezifikationen in die weiteren Phasen der Analyse, des Designs, der Implementierung und des Testens mit einfließen zu lassen und sukzessive die Sicherheitsspezifikation auszubauen. Bedrohungen und Risiken, die bereits in der Geschäftsprozessanalyse erkannt werden, sind in der

Anwendungsfallmodellierung zu erweitern, bei der Systemanalyse in die Ablaufmodellierung einzubeziehen und beim Ausarbeiten des Designs zur Bestimmung von sicherheitsrelevanten Komponenten heranzuziehen. Auch für die Erstellung von Testfällen und bei der Testauswertung dürfen die Bedrohungen mit ihren Verfeinerungen und die im Laufe der Entwicklung weiter erarbeiteten Spezifikationen zur Zugriffssicherheit nicht außer acht gelassen werden.

SPA3: Gemeinsame Entwicklung von Funktion und Sicherheit

Die Modellierung von Funktionen, Daten und Aspekten der Zugriffssicherheit hat gemeinsam zu erfolgen, sodass zusätzlicher Modellierungsaufwand vermieden und gegenseitige Erkenntnisse ausgenutzt werden. Indem in Abläufen und Daten a priori Sicherheitsaspekte modelliert und Abläufe sowie Daten nicht in einem expliziten Schritt an die Sicherheit angepasst werden, wird ein zusätzlicher Modellierungsaufwand vermieden. Es ist bereits in frühen Phasen der Systemmodellierung möglich, Sicherheitseigenschaften abstrakt, d.h., auf einer hohen Modellierungsstufe, einzubeziehen. So kann beispielsweise frühzeitig eine Beweissicherung in einen Ablauf integriert und eine spätere zusätzliche Veränderung des Ablaufs vermieden werden.

Bei der gemeinsamen Daten- und Funktionsmodellierung mit Aspekten der Zugriffssicherheit ergeben sich gegenseitige Erkenntnisse. Analog zur Anwendungsfallmodellierung, bei der Synergien zwischen Daten und Abläufen ermittelt und in der gegenseitigen Modellierung integriert werden, liefert hier die Modellierung der Zugriffssicherheit Erkenntnisse für Verhalten und Daten. Die Modellierung der Daten und des Verhaltens stellt andererseits Input für die Modellierung der Zugriffssicherheit zur Verfügung. Getrieben durch die Betrachtung des Schutzziels *Authentifikation* kann etwa in Web-basierten Anwendungen ein Sitzungskonzept (vgl. hierzu Abschnitt 7.2) eingeführt werden und dies erfordert Änderungen an den Abläufen. Im Gegenzug kann beispielsweise bei der Beschreibung eines Anwendungsfalls erkannt werden, dass zur Ausführung dieser Funktion der Benutzer im System angemeldet sein muss. Aspekte der Zugriffssicherheit und die Funktions- und Datenmodellierung beeinflussen sich gegenseitig, es findet eine so genannte *feature interaction* statt. Die Erkenntnisse zur Lösung sind bei beiden Modellierungen zu betrachten, sodass letztendlich ein Systemverhalten auftritt, das beide Modellierungen abdeckt.

Prozesse zur Entwicklung von zugriffssicheren Systemen müssen so gestaltet werden, dass Entwickler bei allen ihren Aktivitäten zur Modellierung der Funktionalität des Systems Aspekte der Zugriffssicherheit betrachten müssen. So kann gewährleistet werden, dass alle Teile des Systems einer Sicherheitsanalyse unterzogen werden und Maßnahmen zur Abwehr eingeführt werden.

SPA4: Lokale Spezifikation der Sicherheit

Wie in den vorab vorgestellten existierenden Methoden zur Entwicklung zugriffssicherer Systeme gezeigt wurde, werden in den vorgestellten Ansätzen Aspekte der Zugriffssicherheit eines Systems in seinen frühen Phasen immer in seiner Gesamtheit betrachtet. So wird etwa im phasenstrukturierten Modell vor dem Grobentwurf eine Bedrohungs- und Risikoanalyse für das gesamte System durchgeführt und daraus eine Sicherheitsstrategie und ein Sicherheitsmodell entwickelt. Auch beim SEC-Vorgehen wird im Rahmen der System-Anforderungsanalyse eine globale Bedrohungs- und Risikoanalyse für das System durchgeführt. Obwohl bereits die Systeme in funktionale Anforderungen zerlegt sind, beginnt die Sicherheitsanalyse wieder bei der Gesamtheit des Systems. Um sicherzustellen, dass Systemteile in einer Sicherheitsbetrachtung

nicht übersehen und alle funktionellen Anforderungen auch bezüglich der Sicherheit untersucht werden, sind Anforderungen an die Zugriffssicherheit bereits bei der Analyse der funktionalen Anforderungen und des Domänenmodells zu betrachten, sodass eine nachträgliche Aufsplittung der Systemanforderungen zur Analyse der Zugriffssicherheit vermieden werden kann. Sicherlich werden durch diese Vorgehensweise Systemteile mehrfach einer Sicherheitsanalyse unterzogen, aber es kann bei einer Sicherheitsanalyse einer Funktion gegebenenfalls auf die Sicherheitsanalyse innerhalb einer bereits erarbeiteten Funktion verwiesen werden. Wird eine Sicherheitsanalyse mehrfach für funktionale Anforderungen mit gleichen Bedrohungen und Maßnahmen durchgeführt und dies erst später festgestellt, so können die beiden Analysen verglichen werden und wir erhalten dadurch ein Kontrollinstrument zur Validierung. Beispielsweise kann bei der Systementwicklung zum Datenmodell eine Zugriffstabelle erstellt werden, in den Funktionen werden im Rahmen der Verhaltensmodellierung die Rechte der Ein- und Ausgabedaten sowie die Rechte der in der Funktion veränderten Daten des Datenmodells spezifiziert. Die Zugriffsrechte zum Verändern der Daten im Datenmodell sind ebenfalls Bestandteil der Zugriffstabelle und können nun auf Inkonsistenzen überprüft und ggf. korrigiert bzw. ergänzt werden.

SPA5: Betrachtung von Produktartefakten

Neben den Prozessabläufen spielen Produktartefakte, welche die Ein- und Ausgabeparameter der Prozesse darstellen, eine bedeutende Rolle. Produktartefakte sind dabei nicht zwingend Dokumente, denn die Implementierung oder das fertig kompilierte System stellen ebenfalls Produktartefakte dar und ein Dokument kann durchaus auch aus mehreren Produktartefakten bestehen. Produktartefakte sind somit fein granularer als Dokumente. Mit ihnen ist es möglich, die nötige Zustandsmenge zur Ausführung der Prozessaktivität zu beschreiben und dabei abgeänderte und erzeugte Produktartefakte als Ausgaben zu spezifizieren. In neueren Prozessmodellen (vgl. [GMP+03b]) wird in Abhängigkeit der aktuellen Zustandsmenge der Produktartefakte die nächste Aktion ausgewählt, so dass der Prozessablauf dynamisch bestimmt berechnet kann. Eine in den Prozessanforderungen SPA1 und SPA2 geforderte durchgängige und sukzessive Entwicklung der Zugriffssicherheit fordert die Betrachtung von Produktartefakten, in der für jede Aktivität angegeben wird, was bereits gegeben, was erzeugt bzw. verändert wird.

SPA6: Iterative Entwicklung

Risiken frühzeitig identifizieren und reduzieren, eine robuste Architektur erstellen, ändernde Anforderungen betrachten, das System inkrementell aufbauen sowie effektives Arbeiten der Teammitglieder sind nach [JBR99] Gründe, dass heutige objektorientierte Entwicklungsprozesse iterativ und inkrementell arbeiten. Auch im Bereich des Security Engineerings ist es nötig, dass Sicherheitsrisiken frühzeitig identifiziert, reduziert und gegebenenfalls durch die Erstellung eines Prototyps validiert werden. Eine iterative Entwicklung ist notwendig für den Aufbau einer Systemarchitektur mit Trennung der sicherheitskritischen und unkritischen Teile. Zu verändernde funktionale Anforderungen bringen Änderungen in den Anforderungen an die Zugriffssicherheit mit sich, die in weiteren Iterationen behandelt werden müssen. Die Größe von Systemen macht es oftmals unumgänglich, dass Systeme in mehreren Inkrementen entwickelt werden und da durch die Integration der Zugriffssicherheit die Systeme noch komplexer werden, wird die inkrementelle Entwicklung noch mehr gefordert.

Tabelle 5.1: Abdeckung der Prozessanforderungen in Vorgehensmodellen

Anforderung	**Phasen-strukturiert**	**V-Modell und ITSEC**	**CC konformes Vorgehen**
Durchgängige Sicherheitsbetrachtung	+	+	+
Sukzessive Erarbeitung der Sicherheit	+	+	+
Gemeinsame Entwicklung von Funktionen und Sicherheit	–	+	+
Lokale Spezifikation der Sicherheit	–	–	–
Betrachtung von Produktartefakten	–	–	○
Iterative Entwicklung	–	○	–
Sicherheitsmodellierung in der Geschäftsprozessanalyse	–	–	–

Legende: + wird unterstützt ○ wird geringfügig unterstützt – wird nicht unterstützt

Zudem spielt die iterative Entwicklung bei der Entwicklung zugriffssicher Systeme eine bedeutende Rolle, denn Maßnahmen zur Abdeckung der Anforderungen an die Zugriffssicherheit, die innerhalb einer Iteration in das System integriert werden, rufen Änderungen und Ergänzungen hervor, die in weiteren Iterationen vor unerlaubtem Zugriff geschützt werden müssen. So kann etwa eine Sicherheitsmaßnahme zur Beweissicherung eine Speicherung der Änderungen hervorrufen. Die Speicherung der Änderungen führt zu einer zusätzlichen Anforderung, die in einer der nächsten Iterationen ebenfalls gegenüber den Sicherheitsaspekten zu prüfen und umzusetzen ist.

SPA7: Sicherheitsmodellierung in der Geschäftsprozessanalyse

In der Geschäftsprozessanalyse werden zusammenhängende Folgen von Tätigkeiten, die in Unternehmen zur Erreichung der Unternehmens- und Organisationsziele erledigt werden, modelliert [Sta01]. Dabei stehen längere, zusammenhängende Folgen von Tätigkeiten, die zur Erledigung einer Aufgabe notwendig sind, im Mittelpunkt. An der Geschäftsprozessanalyse sind sowohl Anwender, Auftraggeber, Auftragnehmer wie auch Anforderungsentwickler gemeinsam beteiligt. Bei der Ausarbeitung der Anforderungen stoßen sie unumgänglich auf Aspekte der Zugriffssicherheit, beispielsweise steht zu diesem Zeitpunkt bereits fest, dass bestimmte Aktionen nur einem engen Benutzerkreis vorenthalten sind, dass vertrauliche Daten über das Netz ausgetauscht werden müssen oder dass das Verändern von Dokumenten, Verträgen etc., protokolliert werden muss.

In [Bal98] werden Anforderungen an die Sicherheit als abstrakte QoS-Eigenschaften spezifiziert. Obwohl zu diesem Zeitpunkt bereits detailliertes Wissen zur Zugriffssicherheit vorliegt oder ermittelt werden kann, wird dies nicht dokumentiert. Anforderungsentwickler müssen sich in einem eigenen Arbeitsschritt dieses Wissen erarbeiten, oftmals auch ohne Beisein des Kunden. Um dies zu vermeiden, wird ein Vorgehensmodell benötigt, welches das bereits bekannte Wissen innerhalb der Geschäftsprozessanalyse geeignet dokumentiert und ein Instrument zur analytischen Ermittlung der Anforderungen an die Zugriffssicherheit darstellt.

Tabelle 5.2: Abdeckung der Prozessanforderungen in Vorgehensbeschreibungen

Anforderung	**Sicherheit in frühen Phasen**	**Sicherheit im Spiral-modell**	**Security Engineering Management**
Durchgängige Sicherheitsbetrachtung	○	○	○
Sukzessive Erarbeitung der Sicherheit	○	○	○
Gemeinsame Entwicklung von Funktionen und Sicherheit	○	○	○
Lokale Spezifikation der Sicherheit	○	○	–
Betrachtung von Produktartefakten	○	○	○
Iterative Entwicklung	○	○	○
Sicherheitsmodellierung in der Geschäftsprozessanalyse	–	–	–

Legende: ○ wird unterstützt oder erwähnt – wird nicht unterstützt

5.2.2 Bewertung existierender Vorgehensmodelle

In den Tabellen 5.1 und 5.2 sind die Bewertungen der oben vorgestellten existierenden Vorgehensmodelle bzw. -beschreibungen zusammengefasst.

Derzeit existierende Prozessmodelle betrachten Aspekte der Zugriffssicherheit durchgängig, d.h., über allen Phasen hinweg, und die Sicherheitsaspekte werden zudem sukzessive erarbeitet. Weitgehend gemeinsam wird auch der Ansatz verfolgt, dass die Modellierung von Aspekten der Sicherheit gemeinsam mit der funktionalen Modellierung (inklusive der Datenmodellierung) zu erfolgen hat. Die in den Abschnitten 5.1.1 bis 5.1.3 vorgestellten Vorgehensmodelle (vgl. hierzu Tabelle 5.1) betrachten kaum Aspekte einer iterativen Entwicklung und gehen größtenteils auch auf Produktartefakte, die während der Entwicklung zu erstellen sind, nicht ein. Derzeit wird die Sicherheit immer in ihrer Gesamtheit betrachtet, d.h., eine Zerlegung der Sicherheitsaspekte findet nicht wie bei der funktionalen Ausarbeitung statt.

Sicherheit innerhalb der Geschäftsprozessanalyse ist weder in den ausführlich beschriebenen Vorgehensmodellen noch in den weiteren Vorgehensbeschreibungen enthalten. Die neueren Vorgehensbeschreibungen (vgl. Tabelle 5.2) spezifizieren die Sicherheit lokal, d.h., ebenso wie bei der Funktionsmodellierung wird die Sicherheit nicht als Ganzes, sondern als atomare Einheiten analysiert. Es werden neben den Prozessaktivitäten auch die Produktartefakte, die als Ein- und Ausgabeparameter für die Prozessaktivitäten dienen, mitbetrachtet und iterative Aspekte sind ebenfalls Bestandteil dieser Betrachtungen.

5.3 Das Vorgehen

Da unser Fokus in dieser Arbeit auf der Integration von Aspekten der Zugriffssicherheit in den frühen Phasen der Systementwicklung und nicht auf der Entwicklung eines eigenen, gesamten Prozesses liegt, zeigen wir im Folgenden, wie die Aktivitäten zur Entwicklung zugriffssicherer

Systeme in den frühen Phasen einer objektorientierte Methode integriert werden können. Hierzu erweitern wir ein objektorientiertes Vorgehen um eine Anforderungsspezifikation zugriffssicherer Systeme. Im weiteren Verlauf der Arbeit stellen wir die Anforderungsspezifikation zugriffssicherer Systeme im Detail vor.

Zur Beschreibung verwenden wir das in Kapitel 2 vorgestellte Konzept der Prozessmuster. Wir definieren im Folgenden Prozessmuster für die Integration der Anforderungsspezifikation zugriffssicherer Systeme in den Softwarelebenszyklus, für die Phase der Anforderungsspezifikation zugriffssicherer Systeme und für den Sicherheitsmikroprozess.

Während die Anforderungsspezifikation zugriffssicherer Systeme Teil des Lebenszyklus ist, wird der Sicherheitsmikroprozess innerhalb der Anforderungsspezifikation mehrfach angewendet.

5.3.1 Prozessmuster 5.1: Integration der Anforderungsspezifikation zugriffssicherer Systeme in den Softwarelebenszyklus

Kurzbeschreibung

Mit dem Begriff des Softwarelebenszyklus (engl. *Software-Life-Cycle*) verbindet man nach [PB96] die Vorstellung einer Zeitspanne, in der ein Softwareprodukt entwickelt und eingesetzt wird, bis zum Ende seiner Benutzung. Damit verbindet man die Strukturierung dieses Zeitraums in Phasen. Nach [PB96] kann diese Einteilung in Phasen auch beim Einsatz objektorientierter Techniken erhalten bleiben.

Der in dieser Arbeit entworfene Prozess der Anforderungsspezifikation, der in den nächsten Kapiteln detailliert beschrieben wird, ist in einen Softwarelebenszyklus zu integrieren. Dazu weicht der in diesem Prozessmuster vorgestellte Lebenszyklus von bekannten objektorientierten Lebenszyklen insbesondere dahingehend ab, dass die Ausarbeitungsphase (*engl. Elaboration-Phase* nach [JBR99, Kru00]) durch eine Phase zur Anforderungsspezifikation zugriffssicherer Systeme ersetzt wird.

Problem

Die Anforderungsspezifikation zugriffssicherer Systeme, wie sie in Form eines Prozessmusters im Abschnitt 5.3.2 vorgestellt wird, ist innerhalb eines Softwarelebenszyklus zu integrieren, dabei ist sie im Rahmen eines bestehenden Lebenszyklusmodells einzubinden. Da die Modellierung innerhalb der Anforderungsspezifikation zugriffssicherer Systeme objektorientierten Grundkonzepten entspricht, insbesondere wird die Struktur in Form von Klassendiagrammen erarbeitet und das Verhalten wird in Ablaufdiagrammen, Aktivitätsdiagrammen, Zustandsdiagrammen, Anwendungsfällen und Komponentendiagrammen modelliert, ist es sinnvoll, die Anforderungsspezifikation in ein objektorientiertes Vorgehensmodell zu integrieren. Insbesondere durch die Komplexität von Softwareprodukten hat der Softwarelebenszyklus eine iterative Systementwicklung zu unterstützen. Die notwendigen und erzeugten Produktartefakte des Prozesses sind zu spezifizieren.

Lösung

Objektorientierte Vorgehensmodelle wie zum Beispiel der Rational Unified Process (RUP) [Kru00], der Unified Process (UP) [JBR99] oder der Catalysis Approach [DW98] sind anpassbar und erweiterungsfähig. Da sie zumeist sehr allgemein sind, müssen sie vor einer

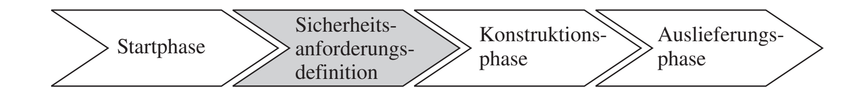


Abbildung 5.4: Übersicht über das modifizierte Lebenszyklusmodell

konkreten Anwendung den Projekt- oder Unternehmensbedürfnissen angepasst werden. Wir wählen hiervon den RUP als Vertreter der objektorientierten Vorgehensmodelle aus und integrieren die Phase der Anforderungsspezifikation zugriffssicherer Systeme in den Softwarelebenszyklus. Der im RUP vorgestellte Lebenszyklus enthält eine Startphase (*engl. Inception Phase*), eine Ausarbeitungsphase (*engl. Elaboration Phase*), eine Konstruktionsphase (*engl. Construction Phase*) und eine Auslieferungsphase (*engl. Transition Phase*). Alle Phasen sind auf eine objektorientierte Modellierung ausgerichtet, Iterationen sind ebenfalls in allen Phasen zu finden und Produktmodelle stehen ebenfalls im Vordergrund.

Da die zu integrierende Phase der Anforderungsspezifikation zugriffssicherer Systeme größtenteils eine Verfeinerung der Ausarbeitungsphase darstellt, ersetzen wir diese Phasen gegeneinander. Somit ergibt sich der in Abbildung 5.4 gezeigte Softwarelebenszyklus. Für die Integration ist eine Modifikation der Startphase durchzuführen. Wir ändern sie dahingehend ab, dass ausgehend von einer Projektidee die Mission des Projekts und die globalen Sicherheitsziele bestimmt werden. Der Start der Erarbeitung der Anforderungen (*engl. Requirements*) wird etwas verzögert in der Phase der Anforderungsspezifikation zugriffssicherer Systeme begonnen, da hierfür eigene iterative Prozesse eingeführt werden.

Die in Abbildung 5.4 grau hinterlegte Sicherheits-Anforderungsanalyse wird detailliert in einem eigenen Prozessmuster in Abschnitt 5.3.2 erläutert.

Aktivitäten

Im Rahmen des vorgestellten Softwarelebenszyklusprozesses mit Integration der Anforderungsspezifikation für zugriffssichere Systeme sind die folgenden Aktivitäten durchzuführen:

- Ausführung der Startphase, in der ausgehend von einer Projektidee die Mission des Projekts und das Sicherheitsziel festgelegt werden;
- Durchführung der Anforderungsspezifikation zugriffssicherer Systeme, wie dies im Prozessmuster 5.2 im Abschnitt 5.3.2 beschrieben wird;
- Konstruktion des Systems mit Design, Implementierung und Test, wie zum Beispiel in [Kru00, JBR99, DW98] vorgestellt;
- Durchführung der Auslieferungsphase, beispielsweise nach [Kru00, JBR99].

Produktartefakte

Die folgende Auflistung beschreibt die Eingaben und Ausgaben zum Softwarelebenszyklus inklusive der speziellen Produktartefakte zur Spezifikation der Sicherheitsaspekte. Falls Standardproduktartefakte um Sicherheitsaspekte erweitert werden, ist dies vermerkt.

Eingabe-Produktartefakt

- Projektidee

Ausgabe-Produktartefakte

- Projektmission und Sicherheitsziel

- Mit Sicherheitsaspekten erweiterte Dokumente der Anforderungsspezifikation zugriffssicherer Systeme (Domänenmodell, Geschäftsprozessmodell, Anwendungsfallmodell, Analysemodell, Bedrohungs- und Risikomodell, Benutzerrechtemodell)
- Designdokument mit Nachweis von Sicherheitseigenschaften und Architekturdokument mit sicherheitsspezifischen Komponenten und Packages
- Implementierungsdokumente mit Implementierung der Subsysteme und der Komponenten
- Dokumentation des Testens mit Testplan, Testmodell, Testergebnissen, Fehlern, Test-Packages und Klassen

Kontext

Zur Ausführung des vorgestellten Lebenszyklus sind zusätzlich weitere Managementaufgaben auszuführen, da diese im Lebenszyklusmodell nicht betrachtet wurden. So sind zum Beispiel Ressourcen- und Budgetplanung sowie das Ausarbeiten der Verträge kein Bestandteil des vorgestellten Prozesses.

Struktur

In der Lösung wurde bereits der Ablauf des Lebenszyklusprozesses angegeben. Abbildung 5.5 zeigt eine detaillierte Betrachtung dieses Ablaufs, indem es die Standardphasen und die Kernarbeitsschritte des Prozessmusters darstellt. Zudem sind auch mögliche Iterationen angegeben.

In der Phase der Sicherheitsanalyse sind Iterationen möglich und notwendig, wie dies in den Prozessmustern zur Anforderungsspezifikation zugriffssicherer Systeme und zum Sicherheitsmikroprozess gezeigt wird. Weiterhin sind auch in der Konstruktion und Auslieferung Iterationen möglich, wie in [Kru00, JBR99] beschrieben.

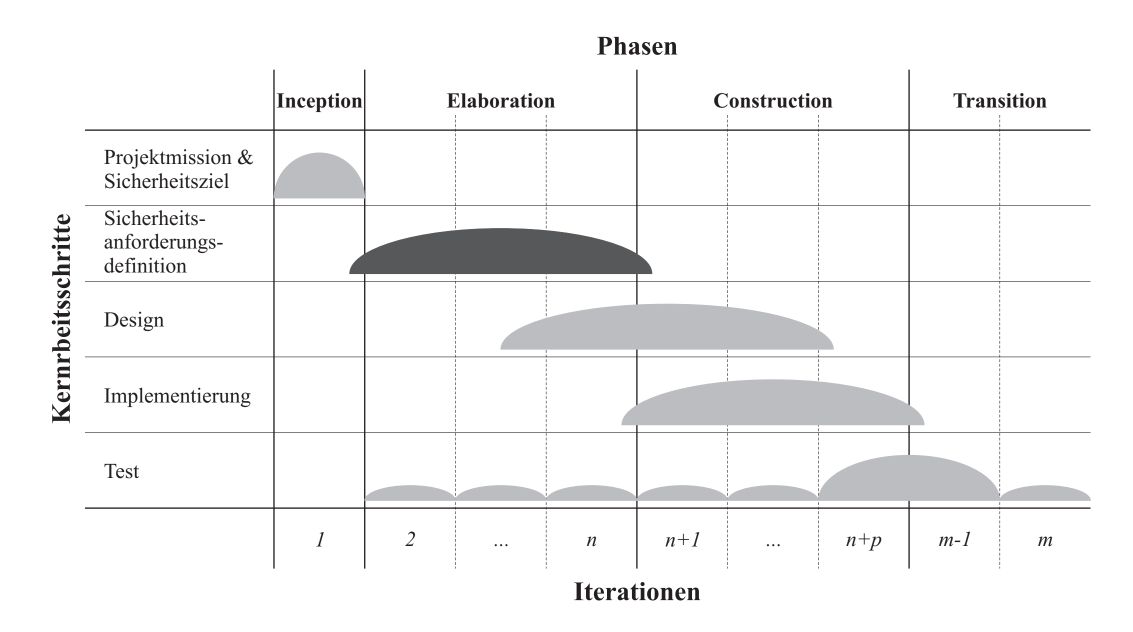


Abbildung 5.5: Phasen, Kernarbeitsschritte und Iterationen im Lebenszyklusmodell

Vor- und Nachteile

Durch die Anwendung des um die Anforderungsspezifikation abgeänderten Lebenszyklusmodells wird eine frühzeitige Betrachtung von Sicherheit im Prozess ermöglicht, da Sicherheitsaspekte in den entsprechenden Prozessabschnitten analytisch erarbeitet werden, wie wir in den einzelnen Prozessmustern noch sehen können.

Anforderungen an die Zugriffssicherheit werden bei der Analyse der funktionalen Anforderungen integriert und umgesetzt, sodass sie im Design und in der Implementierung umgesetzt werden können. Gegebenfalls sind formale Testmethoden anzuwenden, um den Nachweis der Sicherheitserbringung zu garantieren. Da dies jedoch außerhalb des Rahmens dieser Arbeit liegt, gehen wir darauf nicht näher ein.

Soweit Firmen oder Institutionen eigens definierte Prozesse innerhalb der Ausarbeitungsphase festgelegt haben, können diese nicht ohne weiteres gegen eine Phase der Anforderungsspezifikation zugriffssicherer Systeme ausgetauscht werden. In diesen Fällen sind die eigenen Prozesse mit dem Vorgehen in der Anforderungsspezifikation zugriffssicherer Systeme zu kombinieren; der vorgestellte Ansatz ist durch die Anwendung von Prozessmustern jedoch für derartige Anpassungen offen.

In Beziehung stehende Prozessmuster

Auszuführender Teilprozess

- Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme (siehe Abschnitt 5.3.2) ◇

5.3.2 Prozessmuster 5.2: Anforderungsspezifikation zugriffssicherer Systeme

Kurzbeschreibung

Nach der Startphase eines Softwareprojektes, in der Projektmission und Sicherheitsziel geklärt werden, sind die Anforderungen zu definieren, d.h., die Anforderungen sind zu ermitteln, festzulegen und zu beschreiben, zu analysieren und zu verabschieden (vgl. [Bal96]).

Wie bereits erläutert wurde, ist es für die Entwicklung zugriffssicherer Systeme unabdingbar, Sicherheit in der Entwicklung frühzeitig bei der Erarbeitung der Anforderungsspezifikation zu betrachten. Da sich die Ausarbeitung der Anforderungen über mehrere Prozessabschnitte erstreckt und Sicherheitsaspekte jeweils in dem Prozessabschnitt behandelt werden müssen, in der sie ermittelt werden, ist es notwendig, alle Prozessabschnitte der Anforderungsspezifikation mit Aspekten der Zugriffssicherheit anzureichern.

In diesem Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme stellen wir die Prozessabschnitte der Anforderungsspezifikation zugriffssicherer Systeme vor und geben einen kurzen Überblick über die Aktivitäten in den Prozessabschnitten und die Integration der Sicherheitsaspekte. Die einzelnen Abschnitte der Anforderungsspezifikation werden später in den Kapiteln 6 bis 8 der Arbeit detailliert erläutert.

Problem

Damit die Zugriffssicherheit im erforderlichen Maß innerhalb der Entwicklung von zugriffssicheren Systemen betrachtet wird, ist es notwendig, diese frühzeitig, durchgängig und sukzessive in den Entwicklungsprozess zu integrieren. Für die Anforderungsspezifikation bedeutet

dies, dass Sicherheitsaspekte des Systems bereits von Anfang an, d.h., ab der Geschäftsprozessanalyse, betrachtet werden. Für die durchgängige und sukzessive Entwicklung ist es notwendig, dass die in der Geschäftsprozessanalyse erarbeiteten Ergebnisse in allen weiteren Prozessabschnitten der Anforderungsanalyse ergänzt und verfeinert sowie dass die in den jeweiligen Phasen erarbeiteten Ergebnisse in Form von Produktartefakten auch in späteren Prozessabschnitten betrachtet und verwendet werden. Ebenso ist in den frühen Phasen darauf zu achten, dass Sicherheit so weit wie möglich gemeinsam mit der Festlegung der funktionalen Anforderungen stattfindet und dass eine iterative Entwicklung, die bei größeren Entwicklungsprojekten unabdingbar ist, unterstützt wird.

Lösung

Die Analysephase wird in die drei Prozessabschnitte zur Geschäftsprozessmodellierung, zur Anwendungsfallmodellierung und zur Analyse zugriffssicherer Systeme unterteilt. In allen Abschnitten werden Aspekte der Zugriffssicherheit herausgearbeitet und analysiert. Ab dem Abschnitt der Geschäftsprozessmodellierung, in der Geschäftsprozesse zur Modellierung von Unternehmensabläufen in Teilaufgaben zerlegt werden, werden bereits Sicherheitsaspekte ausgearbeitet. So werden beispielsweise im Domänenmodell Schutzziele angegeben oder im Aktivitätsdiagramm die Anforderungen der Zugriffssicherheit an die zu übermittelnden Datenobjekte spezifiziert.

In der Anwendungsfallmodellierung, in der ausgehend von Akteuren das Verhalten, welches in der Geschäftsprozessmodellierung spezifiziert wurde, in Teilaufgaben zerlegt und detailliert analysiert wird, werden die bereits erarbeiteten Sicherheitsziele wieder aufgegriffen und hieraus Anwendungsfälle erarbeitet und beschrieben. Die Sicherheitsanforderungen der lokal betrachteten Funktionalität, d.h., des funktional modellierten Teils des Anwendungsfalls, werden hier neben den funktionalen Anforderungen im Anwendungsfall schrittweise erarbeitet und beschrieben. Das mit Sicherheitsaspekten bereits angereicherte Domänenmodell aus der Geschäftsprozessmodellierung wird verfeinert und die Zugriffsrechte auf die Daten- und Flussobjekte werden innerhalb einer Zugriffsmatrix spezifiziert und prädikativ formalisiert. In der Analyse zugriffssicherer Systeme werden aus den Sicherheitsanwendungsfällen, aus dem mit Sicherheitsaspekten angereichertem Klassendiagramm und aus der Zugriffsmatrix Szenarien modelliert und die identifizierten Sicherheitsanforderungen und -ziele in diesen Szenarien umgesetzt.

Abbildung 5.6 gibt einen Überblick über die Anforderungsspezifikation zugriffssicherer Systeme mit ihren drei Prozessabschnitten.

Für eine schrittweise Erarbeitung der Aspekte der Zugriffssicherheit in den einzelnen Prozessabschnitten wird innerhalb jedes Abschnitts der Sicherheitsmikroprozess ausgeführt, in dem die Schutzziele erarbeitet, die Bedrohungen modelliert und die Risiken bewertet sowie Maßnahmen erarbeitet und überprüft werden. Eine genaue Beschreibung des Sicherheitsmikroprozesses wird im Prozessmuster zum Sicherheitsmikroprozess im Abschnitt 5.3.3 gegeben.



Abbildung 5.6: Die Phasen der Anforderungsspezifikation zugriffssicherer Systeme

Die mehrmalige Anwendung des Sicherheitsmikroprozesses innerhalb der einzelnen Phasen führt zu einer iterativen Erarbeitung der Anforderungen an die Zugriffssicherheit. Besonders im Bereich der Sicherheitsmodellierung ist eine derartige iterative Vorgehensweise von besonderer Bedeutung, da spezifizierte funktionale oder strukturelle Eigenschaften oftmals durch gegenseitige Einwirkung oder durch Einbringung von Sicherheitsaspekten abgeändert werden müssen und somit die geänderte Modellierung erneut einer Sicherheitsanalyse unterzogen werden muss. So kann z.B. die Teilung eines Datenobjekts des Domänenmodells zu einer neuen Ausprägung der Zugriffsmatrix führen oder die Einführung eines Objekts zur Beweissicherung das Domänenmodell erweitern. Beide Änderungen erfordern eine erneute Sicherheitsanalyse.

Aktivitäten

Im Rahmen der Prozessausführung für die Anforderungsspezifikation zugriffssicherer Systeme sind die folgenden Aktivitäten nach der in der Lösung skizzierten Vorgehensweise auszuführen:

- Durchführung einer Geschäftsprozessmodellierung zugriffssicherer Systeme (siehe Prozessmuster 7.1 in Abschnitt 6.4.1)
- Durchführung einer Anwendungsfallmodellierung zugriffssicherer Systeme (siehe Prozessmuster 8.1 in Abschnitt 7.5.1)
- Durchführung einer Analyse zugriffssicherer Systeme (siehe Prozessmuster 9.1 im Abschnitt 8.4.1)
- Innerhalb der einzelnen Prozessabschnitte ist jeweils der im Prozessmuster 5.3 (vgl. Abschnitt 5.3.3) vorgestellte Sicherheitsmikroprozess auszuführen.

Produktartefakte

Die folgende Auflistung beschreibt die Eingaben und Ausgaben zum Prozess der Anforderungsspezifikation zugriffssicherer Systeme. Um Sicherheitsaspekte erweiterte Standardproduktartefakte sind gekennzeichnet.

Eingabe-Produktartefakte

- Projektidee
- Projektmission und Sicherheitsziel (aus der Startphase)

Ausgabe-Produktartefakte

- Domänenmodell (mit Sicherheitsaspekten erweitert)
- Geschäftsprozessmodell (mit Sicherheitsaspekten erweitert)
- Anwendungsfallmodell (mit Sicherheitsaspekten erweitert)
- Analysemodell (mit Sicherheitsaspekten erweitert)
- Bedrohungs- und Risikomodell
- Benutzerrechtemodell

Kontext

Das Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme wird in einen objektorientierten Vorgehensprozess, zum Beispiel aus [Kru00, JBR99], integriert. Die Integration der Anforderungsspezifikation zugriffssicherer Systeme in einen Softwarelebenszyklus wurde bereits im Prozessmuster 5.1 in Abschnitt 5.3.1 vorgestellt.

Neben den vorgestellten Aufgaben sind auch Aktivitäten zur Ressourcen- und Budgetplanung durchzuführen, die jedoch im Rahmen dieses Prozessmusters nicht betrachtet werden. Da sich

die vorliegende Arbeit auf Prozesse und Produktartefakte in der Entwicklung zugriffssicherer Software beschränkt, werden derartige Managementaspekte nicht behandelt.

Struktur

Das Aktivitätsdiagramm in Abbildung 5.7 zeigt den Ablauf der Anforderungsspezifikation zugriffssicherer Systeme mit Einbindung des Sicherheitsmikroprozesses, wie dies in der Lösung vorgeschlagen wurde.

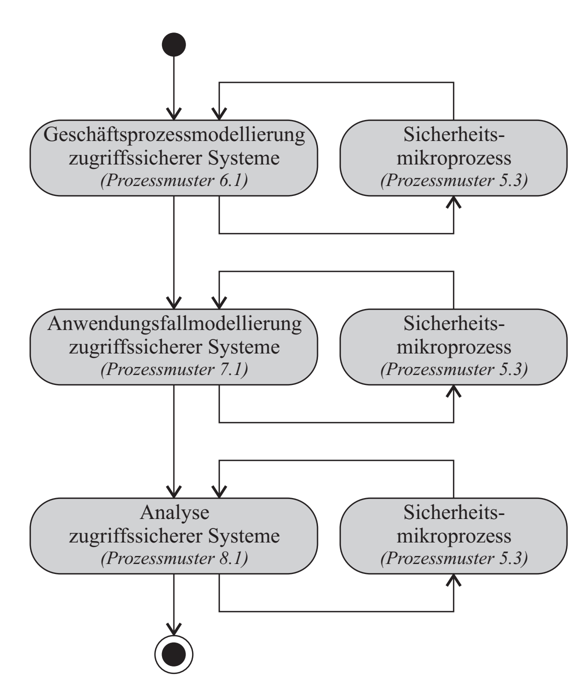


Abbildung 5.7: Der Prozess der Anforderungsspezifikation zugriffssicherer Systeme

Dadurch, dass der Sicherheitsmikroprozess in jedem Prozessabschnitt mehrfach ausgeführt werden kann, wird eine iterative Entwicklung der Anforderungen an die Zugriffssicherheit ermöglicht. Der Sicherheitsmikroprozess kann in jedem Prozessabschnitt so oft wiederholt, bis sich beim Ausführen des Sicherheitsmikroprozesses keine Änderungen an den Sicherheitsaspekten für die jeweilige Stufe der Modellierung ergeben, d.h., dass bezüglich Funktionalität, Sicherheit und Struktur ein lokal stationärer Zustand erreicht wird. In der darauf folgenden Modellierung treten durch Verfeinerung neue funktionale und sicherheitsspezifische Aspekte auf, sodass weitere Iterationen mithilfe des Sicherheitsmikroprozesses notwendig werden. Eine Ausnahme stellt hier nur die Analysemodellierung dar, denn nach deren Abarbeitung müssen alle funktionalen und sicherheitsspezifischen Anforderungen integriert sein, sodass anschließend mit dem Design fortgefahren werden kann.

Vor- und Nachteile

Durch das vorgestellte Vorgehen wird eine iterative Entwicklung von Funktionalität und Aspekten der Zugriffssicherheit ermöglicht, in der Sicherheitsaspekte durchgängig und sukzessive von Anfang an, d.h., ab der Phase der Geschäftsprozessmodellierung, betrachtet werden.

Der vorgestellte Prozess ist in bestehende Vorgehensmodelle integrierbar, insbesondere in objektorientierte Vorgehensweisen, da unter anderem die verwendeten und ergänzten Beschrei-

bungstechniken an denen der objektorientierten Vorgehensweisen [Kru00, DW98, JBR99] angeglichen sind.

Das vorgestellte Prozessmuster beschränkt sich auf eine Neuentwicklung von zugriffssicherer Software. Für die Phase der Anforderungsspezifikation bedeutet dies, dass die Anforderungen gemeinsam mit der Funktionalität und aus einer Bedrohungs- und Risikoanalyse heraus entwickelt werden. Eine Analyse bestehender Systeme, insbesondere von Sicherheitsaspekten, ist nicht Bestandteil dieses Prozessmusters. Hierzu wäre das vorgestellte Prozessmuster mit den zugehörigen Teilphasen abzuändern, was jedoch nicht Gegenstand der vorliegenden Arbeit ist.

In Beziehung stehende Prozessmuster

Auszuführende Teilprozesse

- Prozessmuster zur Geschäftsprozessmodellierung zugriffssicherer Systeme (siehe Abschnitt 6.4.1)
- Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme (vergleiche Abschnitt 7.5.1)
- Prozessmuster zur Analyse zugriffssicherer Systeme (siehe Abschnitt 8.4.1)

Übergeordneter Prozess

- Prozessmuster zur Integration der Anforderungsspezifikation zugriffssicherer Systeme in den Softwarelebenszyklus (siehe Abschnitt 5.3.1)

Weiteres referenziertes Prozessmuster

- Prozessmuster zum Sicherheitsmikroprozess (siehe Abschnitt 5.3.3) ◇

5.3.3 Prozessmuster 5.3: Sicherheitsmikroprozess

Kurzbeschreibung

Innerhalb der Phasen der Anforderungsspezifikation zugriffssicherer Systeme stehen wir neben der Erarbeitung des Verhaltens und der Struktur vor der zentralen Aufgabe, die Anforderungen an die Zugriffssicherheit zu erarbeiten. Grundsätzlich sind dabei Bedrohungen und Risiken systematisch zu erarbeiten, Maßnahmen zu entwickeln und diese zu überprüfen. Da die Modellierung jedoch innerhalb der Phasen der Anforderungsspezifikation verfeinert wird und dadurch die Sicherheit auf unterschiedlichen Detaillierungsstufen zu untersuchen ist, führen wir in diesem Prozessmuster einen Sicherheitsmikroprozess ein, der auf allen Ebenen der Sicherheitsmodellierung innerhalb der Anforderungsspezifikation ausgeführt werden kann.

Problem

Für die schrittweise Erarbeitung von Aspekten der Zugriffssicherheit gemeinsam mit der Funktionalität haben wir im Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme im Abschnitt 5.3.2 eine dreiteilige Anforderungsspezifikation vorgestellt. Innerhalb jedes Prozessabschnitts der Anforderungsspezifikation benötigen wir einen Prozess zur Erarbeitung der Anforderungen an die Zugriffssicherheit, der innerhalb eines Prozessabschnitts auch mehrfach

ausgeführt werden kann. Eine iterative Entwicklung der Aspekte der Zugriffssicherheit innerhalb eines Prozessabschnitts ist notwendig, da durch die Ergänzung von funktionalen Aspekten und Aspekten der Zugriffssicherheit strukturelle und verhaltensorientierte Teile ergänzt werden, die noch keiner Sicherheitsmodellierung unterzogen wurden und somit in einer weiteren Iteration nach Aspekten der Zugriffssicherheit zu untersuchen sind. Da die Prozessschritte für eine Entwicklung zugriffssicherer Systeme innerhalb der verschiedenen Prozessabschnitte gleich sind und nur auf verschiedenen Produktartefakten arbeiten, benötigen wir einen universellen Sicherheitsmikroprozess, der iterativ einsetzbar sein muss.

Lösung

Derzeitige Vorgehensweisen wie beispielsweise [Eck03, And01] zur Erarbeitung von Aspekten der Zugriffssicherheit sehen eine Analyse der Bedrohungen, eine Bewertung der Risiken und einen Entwurf von Maßnahmen vor. Dieser Prozess hat sich in der Entwicklung sicherer Systeme bewährt und wir verwenden diesen als Basisprozess für unseren Sicherheitsmikroprozess.



Abbildung 5.8: Die Aktivitäten des Sicherheitsmikroprozesses

Dadurch, dass wir die Sicherheit nicht in seiner Gesamtheit untersuchen, sondern lokal gemeinsam mit der funktionellen Entwicklung betrachten, verwenden wir keinen Matrix- oder Baum-orientierten Ansatz zur Bedrohungsanalyse (vgl. [Eck03]), der das System für eine Sicherheitsanalyse strukturiert. Anstelle dieser Strukturierung betrachten wir vor der Bedrohungsanalyse in dem lokal behandelten strukturellen oder funktionalen Teil Schutzziele und analysieren daraus Verletzungen dieser Schutzziele, d.h., inwiefern Verletzungen der Vertraulichkeit, Authentizität, Integrität und/oder Verbindlichkeit möglich sind. Aus diesen potenziellen Angriffsmöglichkeiten erarbeiten wir Bedrohungen, Risiken und Maßnahmen.

Nach der Erarbeitung der Maßnahmen führen wir noch einen Schritt zur Überprüfung der Maßnahmen ein, wobei die gewählten Maßnahmen den ermittelten verletzten Sicherheitszielen gegenübergestellt werden. Sollte sich – formal oder informal – zeigen, dass die gewählten Maßnahmen das geforderte Maß an Zugriffssicherheit nicht erfüllen, so sind wiederholt geeignete Maßnahmen zu suchen und auf ihre Tauglichkeit zu überprüfen.

Abbildung 5.8 zeigt die Aktivitäten des Sicherheitsmikroprozesses. Die grau hinterlegten Aktivitäten stellen die Basisaktivitäten zur Ermittlung von Bedrohungen, Risiken und Maßnahmen dar. Die weiß hinterlegten Aktivitäten zeigen die neu eingeführten Aktionen zur Analyse zugriffssicherer Systeme.

Aktivitäten

Bei der Prozessausführung des Sicherheitsmikroprozesses sind die folgenden Aktivitäten nach der in der Lösung und in der Struktur skizzierten Vorgehensweise auszuführen:

- *Untersuchung der Schutzziele*
  In den verschiedenen Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme sind die aktuell zu erstellenden und zu verändernden Produktartefakte bezüglich der Schutzziele Vertraulichkeit, Verbindlichkeit, Integrität und Authentizität

zu untersuchen. So können etwa innerhalb der Geschäftsprozessmodellierung vertrauliche Daten identifiziert oder Aktivitäten, für die vorab eine Authentifikation notwendig ist, ermittelt werden. Werden die Schutzziele innerhalb der einzelnen Prozessabschnitte der Anforderungsspezifikation zugriffssicherer Systeme in einem eigenen Arbeitsschritt erarbeitet, so kann diese Aktivität übersprungen werden. Besondere Bedeutung hat die Ermittlung der Schutzziele im Sicherheitsmikroprozess vor allem bei der iterativen Anwendung, da dabei die Schutzziele vorab in keiner eigenen Prozessaktivität des jeweiligen Prozessabschnitts ermittelt wurden.

- *Modellierung der Bedrohungen*
  Für die innerhalb der vorausgehenden Aktivität gefundenen Schutzziele sind Bedrohungen zu modellieren, d.h., die Gefährdungsbereiche sowie die potenziellen organisatorischen, technischen und benutzerbedingten Ursachen sind systematisch zu ermitteln (siehe [Eck03]). Hier sind beispielsweise Gefährdungsbereiche von vertraulichen Daten genauer zu spezifizieren.
- *Bewertung der Risiken*
  Nicht für jede Bedrohung, die ermittelt wurde, sind umfangreiche Maßnahmen zur Abwendung notwendig. Ausschlaggebend für die zu ergreifenden Maßnahmen sind die Eintretenswahrscheinlichkeiten und der potenzielle Schaden der Bedrohungen, welche bei der Bewertung der Risiken ermittelt werden. Nach [Eck03, HB03] ergeben sich die Eintrittswahrscheinlichkeiten aus dem geschätzten Aufwand, den ein Angreifer zur Durchführung eines Angriffes treiben muss und einer Einschätzung des möglichen Nutzens, den ein Angreifer aus einem erfolgreichen Angriff ziehen könnte.
- *Entwurf von Maßnahmen*
  Abhängig von dem Ergebnis der Risikobewertung sind für die Bedrohungen mit nicht vermeidbarem Risiko Maßnahmen zu entwerfen, die diesen entgegenwirken. Die Maßnahmen sind nach [BBHP05, BBH+03] in die Produktartefakte des jeweiligen Prozessabschnitts (Geschäftsprozess-, Anwendungsfallmodellierung oder Analyse) zu integrieren.
- *Überprüfung der Maßnahmen*
  Nachdem die Maßnahmen entworfen und in die Produktartefakte integriert worden sind, ist bereits innerhalb der Anforderungsspezifikation eine informale, semiformale oder auch formale Überprüfung der Zielführung der Maßnahmen durchzuführen.

Auf Besonderheiten bei der Ausführung des Sicherheitsmikroprozesses in den einzelnen Prozessabschnitten gehen wir bei der Beschreibung der Prozessabschnitte in den Kapiteln 6 bis 8 näher ein. Zudem werden die Bearbeitungsschritte an der im Kapitel 3 eingeführten Fallstudie *TimeTool* exemplarisch gezeigt.

Produktartefakte

Im Gegensatz zu den in den Abschnitten 5.3.1 und 5.3.2 vorgestellten Prozessmustern zum Softwarelebenszyklus und zur Anforderungsspezifikation zugriffssicherer Systeme können wir zu diesem Prozessmuster die Menge der Ein- und Ausgabe-Produktartefakte nicht präzise trennen, da das vorgestellte Prozessmuster auf verschiedenen Stufen der Modellierung zugriffssicherer Systeme angewendet wird und somit unterschiedliche Produktartefakte als Ein- und Ausgabe für den Prozess dienen. Bedingt durch die Verfeinerung stellen zumeist die

Ausgabe-Produktartefakte der Modellierungsstufe n die Eingabe-Produktartefakte der Stufe $n + 1$ dar, sodass sich in dem übergeordneten Sicherheitsmikroprozess Ein- und Ausgabe-Produktartefakte decken.

Ein- und Ausgabe-Produktartefakte

- Geschäftsprozessmodell
- Anwendungsfallmodell
- Analysemodell
- Domänenmodell
- Bedrohungs- und Risikomodell
- Benutzerrechtemodell

Kontext

Da eine Bedrohungs- und Risikoanalyse in den frühen Phasen einer Entwicklung zugriffssicherer Systeme durchzuführen ist, eignet sich der vorgestellte Sicherheitsmikroprozess zur Anwendung innerhalb der im Abschnitt 5.3.2 vorgestellten Anforderungsspezifikation zugriffssicherer Systeme. Durch die Erarbeitung der Sicherheitsanforderungen im mehrmaligen Ausführen des Mikroprozesses eignet sich dieser besonders für mehrphasige Anforderungsspezifikationen und für iterative Vorgehensweisen, wie sie beispielsweise in heutigen objektorientierten Vorgehensmodellen (vgl. [DW98, JBR99, Kru00]) zu finden sind. Weiterhin ist der Sicherheitsmikroprozess durch die aufgeführten Produktartefakte, insbesondere durch das Geschäftsprozessmodell, für betriebliche Informationssysteme geeignet.

Struktur

Abbildung 5.9 gibt den genauen Ablauf der in der Lösung spezifizierten Durchführung des Sicherheitsmikroprozesses wieder. Die vorgestellten Aktivitäten des Mikroprozesses werden hier in einen konkreten Ablauf integriert. Wie bereits erwähnt wurde, sind nach dem Entwurf der Maßnahmen diese zu testen, um sicherzustellen, dass den Bedrohungen mit ihrem ermittelten Risiko entgegengewirkt wird. Falls die Maßnahmen nicht zum Ziel führen, müssen gegebenenfalls neue Maßnahmen ermittelt und diese erneut einer Überprüfung unterzogen werden. Erst nach Findung der geeigneten Maßnahmen terminiert der Sicherheitsmikroprozess und es kann mit der Modellierung fortgefahren werden.

Vor- und Nachteile

Der vorgestellte Sicherheitsmikroprozess unterstützt durch seine Anwendung in allen Prozessabschnitten der Anforderungsspezifikation eine durchgängige und sukzessive Entwicklung von Aspekten der Zugriffssicherheit bereits ab der Phase der Geschäftsprozessmodellierung. Der Sicherheitsmikroprozess ist iterativ anwendbar.

Durch die Bindung an die vorgestellten Produktartefakte ist der Sicherheitsmikroprozess an objektorientierte Beschreibungstechniken ausgerichtet. Grundsätzlich könnte jedoch die vorgestellte Vorgehensweise des Mikroprozesses auch in der funktionalen Modellierung angewendet werden, jedoch wären hierzu einige Abänderungen am Prozessmuster nötig. Da wir uns im Rahmen dieser Arbeit auf objektorientierte Vorgehensweisen und Beschreibungstechniken beschränken, gehen wir auf diese Abänderungen nicht weiter ein.

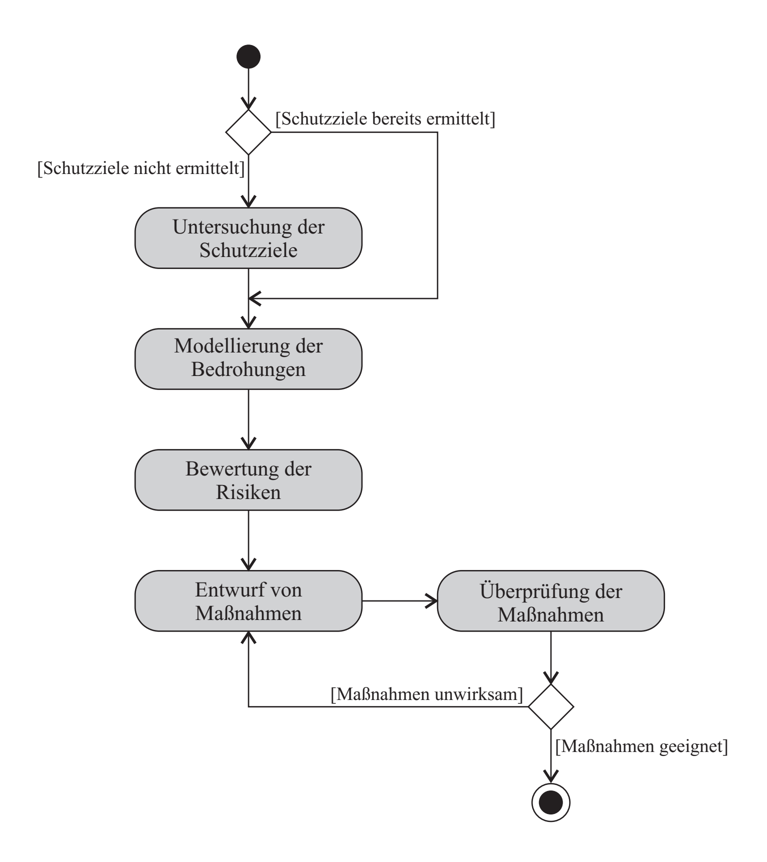


Abbildung 5.9: Der Ablauf des Sicherheitsmikroprozesses

Das Anwendungsgebiet des Sicherheitsmikroprozesses ist durch das Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme im Abschnitt 5.3.2 vorgegeben, das eine gemeinsame Entwicklung von Funktionalität und Sicherheit unterstützt. Der vorgestellte Sicherheitsmikroprozess kann somit nicht in einer ganzheitlichen Entwicklung der Sicherheit eingesetzt werden, da hierbei eine Aufteilung der Sicherheit des Systems in kleinere, handhabbarere Einheiten notwendig wäre, die vom vorgestellten Prozess nicht unterstützt wird.

In Beziehung stehende Prozessmuster

Übergeordnete Prozesse

- Prozessmuster zur Geschäftsprozessmodellierung zugriffssicherer Systeme (siehe Abschnitt 6.4.1)
- Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme (siehe hierzu Abschnitt 7.5.1)
- Prozessmuster zur Analyse zugriffssicherer Systeme (siehe Abschnitt 8.4.1)

Weiteres referenziertes Prozessmuster

- Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme (siehe Abschnitt 5.3.2) ◇

5.4 Übersicht über die Produktartefakte

In den vorgestellten Prozessmustern wurden bereits die Produktartefakte, welche die Ein- und Ausgaben zu den Prozessmustern darstellen, erläutert. Da unser Fokus auf der Anforderungsspezifikation zugriffssicherer Systeme liegt, geben wir im Folgenden einen Überblick über Produktartefakte der Anforderungsspezifikation und zeigen die Zusammenhänge dieser Produktartefakte.

Im Rahmen der Anforderungsspezifikation zugriffssicherer Systeme sind die folgenden Produktartefakte zu erstellen. Soweit es sich um keine eigenen Modelle zur Zugriffssicherheit handelt, sind diese für eine Spezifikation von Aspekten der Zugriffssicherheit anzupassen, wie detailliert in den Kapiteln 6 bis 8 gezeigt wird.

Domänenmodell. Die Struktur des Anwendungskerns wird in einem Klassendiagramm modelliert. Dieses Klassendiagramm wird in der Geschäftsprozessmodellierung und in der Anwendungsfallmodellierung um Datenobjekte ergänzt. Da es sich hierbei um ein klassisches Modell handelt, sind Aspekte der Zugriffssicherheit im Klassendiagramm zu ergänzen.

Geschäftsprozessmodell. Die im Betrieb durchgeführten Aufgaben werden in Abläufen, d.h., in Geschäftsprozessen, modelliert (vgl. [Sta01, EP00]). Die Menge aller Geschäftsprozesse mit den beteiligten Akteuren sowie eine Beschreibung von Architektur, Regeln, Glossar und Ergänzungen bilden das Geschäftsprozessmodell. Dieses klassische Modell muss ebenfalls um Aspekte der Zugriffssicherheit ergänzt werden.

Anwendungsfallmodell. Die einzelnen Aktivitäten in den Geschäftsprozessen, die computergestützt ausgeführt werden können, werden auf Anwendungsfälle des Systems (vgl. [JCJO92, JBR99, Kru00, Bre01]) abgebildet. Die Menge aller Anwendungsfälle mit UI-Prototyp und Ergänzungen bilden das Anwendungsfallmodell, welches ebenfalls um Aspekte der Zugriffssicherheit erweitert werden muss.

Analysemodell. Exemplarische Szenarien (vgl. [Bre01]) für das in den Anwendungsfällen beschriebene Verhalten, ein Analyseklassenmodell, die Analyse-Packages und die Spezialanforderungen bilden das Analysemodell. Das Analysemodell ist um Aspekte der Zugriffssicherheit zu ergänzen.

Bedrohungs- und Risikomodell. Dieses spezielle Modell zur Spezifikation der Zugriffssicherheit beschreibt die Bedrohungen und Risiken, Maßnahmen sowie die Überprüfungen der Maßnahmen, die bei der Modellierung der Zugriffssicherheit durchzuführen sind. Dieses Modell ist eine Sammlung der Ergebnisse aus den Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme.

Benutzerrechtemodell. Die Benutzerrechte (vgl. Kapitel 4) aus der Geschäftsprozessmodellierung, der Anwendungsfallmodellierung und der Analyse sind im Benutzerrechtemodell zu beschreiben. Das Benutzerrechtemodell ist kein Bestandteil der klassischen Anforderungsspezifikation.

Tabelle 5.3 gibt einen Überblick über die Produktartefakte der Anforderungsspezifikation zugriffssicherer Systeme inklusive der hinzuzufügenden Inhalte in den jeweiligen Prozessabschnitten der Anforderungsspezifikation. Wie bereits erläutert, sind das Domänenmodell, das

Tabelle 5.3: Übersicht über die Produktartefakte der Anforderungsspezifikation zugriffssicherer Systeme

	Geschäftsprozessmodellierung zugriffssicherer Systeme	**Anwendungsfallmodellierung zugriffssicherer Systeme**	**Analyse zugriffssicherer Systeme**
Domänenmodell	Mit Aspekten der Zugriffssicherheit erweitertes Klassendiagramm des Anwendungskerns.	An die Systemgrenzen angepasstes und erweitertes Klassendiagramm des Anwendungskerns.	—
Geschäftsprozess-modell	Mit Aspekten der Zugriffssicherheit erweiterte Aktivitätsdiagramme und hierarchisch angeordnetes Modell der Akteure. Glossar enthält grundlegende Begriffe.	An die Systemgrenzen angepasste Geschäftsprozesse und Modellierung der Akteure. Erweitertes Glossar.	—
Anwendungsfall-modell	—	Sicherheitsanwendungsfälle für Sicherheitsgrundfunktionen und mit Aspekten der Zugriffssicherheit erweiterte Anwendungsfälle.	Verfeinerte Anwendungsfall-beschreibung, falls Beschreibung zur Transformation in Szenarien zu ungenau.
Analysemodell	—	—	Klassendiagramm der Analyse mit Vorgangs-, Interaktions- und Fachklassen. Realisierung der Anwendungsfälle in Szenarien. Auswahl und Definition von Protokollen.
Bedrohungs- und Risikomodell	Beschreibung der Bedrohungen, Risiken, Maßnahmen und der Überprüfungen der Maßnahmen.	Ergänzung und Anpassung der Bedrohungen, Risiken, Maßnahmen und der Überprüfungen der Maßnahmen.	Ergänzung und Anpassung der Bedrohungen, Risiken, Maßnahmen und der Überprüfungen der Maßnahmen.
Benutzerrechte-modell	Informale oder prädikative Beschreibung der Zugriffsrechte auf Objekte des Anwendungskerns und der Ausführungsrechte auf Aktivitäten der Geschäftsprozesse in einer Benutzerrechtematrix.	Erweiterung und Verfeinerung der informal oder prädikativ beschriebenen Zugriffsrechte auf Objekte des Anwendungskerns und der Ausführungsrechte auf Anwendungsfälle.	Verfeinerung und Ergänzung der Zugriffs- und Ausführungsrechte. Alle Benutzerrechte sind in der Benutzerrechtematrix prädikativ formalisiert.

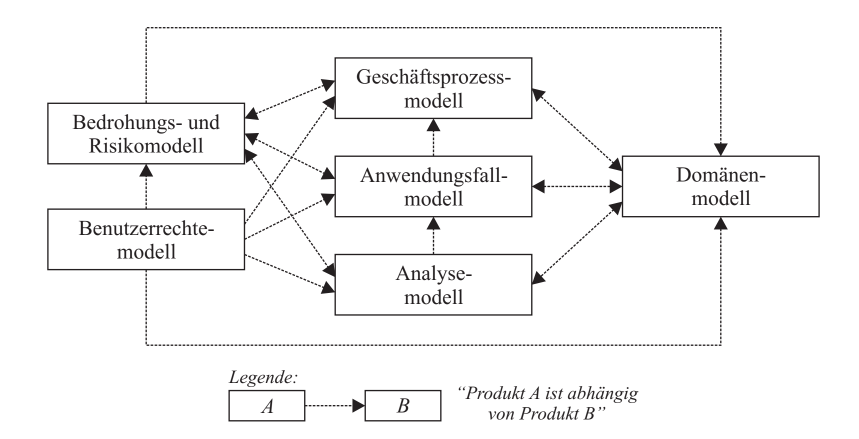


Abbildung 5.10: Produktartefakte der Anforderungsspezifikation mit ihren Abhängigkeiten

Geschäftsprozessmodell, das Anwendungsfallmodell und das Analysemodell Bestandteil einer herkömmlichen objektorientierten Entwicklung. Diese Modelle müssen an die Entwicklung zugriffssicherer Systeme angepasst werden. Beim Bedrohungs- und Risikomodell sowie beim Benutzerrechtemodell handelt es sich um eigenständige Modelle zur Spezifikation der Zugriffssicherheit.

Produktartefakte sind die Ein- und Ausgaben für die Prozessmuster – bestimmte Produktartefakte müssen für die Ausführung des Prozessmusters vorhanden sein, sodass ein Prozessmuster abgearbeitet werden kann. Dabei werden Ausgabe-Produktartefakte erzeugt. Derartige Abhängigkeiten zwischen den Produktartefakten können in einem Abhängigkeitsgrafen modelliert werden. Abbildung 5.10 zeigt die Produktartefakte mit ihren Abhängigkeiten als Grafen. Bei der Erarbeitung des Geschäftsprozessmodells wird das Domänenmodell sowie das Bedrohungs- und Risikomodell gemeinsam und unter gegenseitigem Einfluss erarbeitet, ebenso das Anwendungsfallmodell und das Analysemodell, nur dass hierbei die bereits erarbeiteten Ergebnisse aus dem Geschäftsprozessmodell bzw. aus dem Anwendungsfallmodell mit einfließen.

Das Benutzerrechtemodell, das wie in Tabelle 5.3 gezeigt, innerhalb der Geschäftsprozessmodellierung, der Anwendungsfallmodellierung und der Analyse entwickelt wird, hängt von allen anderen Produktartefakten der Anforderungsspezifikation zugriffssicherer Systeme ab, ist aber keine zwingende Voraussetzung für die Erarbeitung der anderen Produktartefakte.

Die Beziehungen zwischen den Produktartefakten werden bei der Diskussion der Prozessabschnitte der Anforderungsspezifikation zugriffssicherer Systeme in den Kapiteln 6 bis 8 diskutiert.

5.5 Abdeckung der Prozessanforderungen

Zu Beginn dieses Kapitels haben wir im Abschnitt 5.1 existierende Vorgehensmodelle zur Entwicklung sicherer Systeme betrachtet und aus den Bewertungen dieser Vorgehensmodelle Prozessanforderungen zur Entwicklung zugriffssicherer Systeme (vgl. Abschnitt 5.2) abgeleitet. Nachdem wir im vorausgehenden Abschnitt 5.3 das Vorgehen für eine Anforderungsspezifikation zugriffssicherer Systeme vorgestellt haben, untersuchen wir im Folgenden für den vorgestellten Prozess die Abdeckung der Prozessanforderungen. Wie bereits erwähnt, sind

die vorgestellten Prozessanforderungen für einen gesamten Prozess zur Entwicklung sicherer Systeme gültig. Da wir uns im Rahmen dieser Arbeit auf die Phase der Anforderungsspezifikation zugriffssicherer Systeme beschränken, bewerten wir nur für diesen Ausschnitt die Abdeckung der Prozessanforderungen.

Durchgängige Sicherheitsbetrachtungen (SPA1)

Ein zentrales Problem bei der Entwicklung von zugriffssicheren Systemen liegt darin, dass Sicherheitsbetrachtungen oftmals erst zu spät in die Entwicklung mit einbezogen werden oder dass Sicherheit nicht in allen Entwicklungsphasen Betrachtung findet. Im vorgestellten Prozess zur Anforderungsspezifikation zugriffssicherer Systeme werden Aspekte der Zugriffssicherheit bereits ab der Geschäftsprozessmodellierung erarbeitet und in den Prozessabschnitten der Anwendungsfallmodellierung und der Analyse erweitert und verfeinert. Die Durchgängigkeit der Entwicklung der Zugriffssicherheit ist erfüllt, denn in allen Prozessabschnitten der Anforderungsspezifikation werden Sicherheitsaspekte herausgearbeitet und verfeinert.

Sukzessive Erarbeitung von Sicherheit (SPA2)

Eine durchgängige Ausarbeitung der Sicherheitsaspekte in allen Teilphasen reicht nicht aus. Wie Anderson in [And01] beschreibt, werden die Ergebnisse, die innerhalb einer Phase produziert werden, oftmals beiseite gelegt und in der nächsten Phase zur Modellierung nicht herangezogen. Neben der durchgängigen Entwicklung ist es somit notwendig, dass die Sicherheitsaspekte in verschiedenen Phasen und auch innerhalb der Phasen der Anforderungsspezifikation zugriffssicherer Systeme sukzessive ermittelt werden, d.h., dass Ergebnisse des vorausgehenden Prozessabschnitts in den nächsten Prozessabschnitt einfließen. Wie aus der Tabelle 5.3 zur Übersicht über die Produktartefakte und aus der Abbildung 5.10 zu den Produktartefakten und deren Abhängigkeiten hervorgeht, werden innerhalb der Anforderungsspezifikation zugriffssicherer Systeme die Produktartefakte sukzessive erstellt. Das Vorgehen ist auf eine sukzessive Entwicklung ausgerichtet. Auf Details der sukzessiven Entwicklung gehen wir auch innerhalb der detaillierten Betrachtung der Teilphasen in den Kapiteln 6 bis 8 ein.

Gemeinsame Entwicklung von Funktion und Sicherheit (SPA3)

Innerhalb aller Prozessabschnitte der Anforderungsspezifikation zugriffssicherer Systeme steht die gemeinsame Entwicklung von Funktion und Aspekten der Zugriffssicherheit im Vordergrund. Wie bereits skizziert wurde und detailliert in den Kapiteln 6 bis 8 beschrieben ist, findet beispielsweise innerhalb der Geschäftsprozessmodellierung neben der funktionalen und strukturellen Spezifikation auch eine Modellierung des Zugriffs auf Datenobjekte des Domänenmodells und auf Flussobjekte statt. Die Anwendungsfälle werden ebenfalls mit Aspekten der Zugriffssicherheit angereichert und in der Analyse finden wir bei den Interaktionen Nachrichten zur Abdeckung der Zugriffssicherheit. Weiterhin wird die gemeinsame Entwicklung von Funktion und Sicherheit auch durch den eingeführten Sicherheitsmikroprozess unterstützt, denn beliebig große funktionelle Einheiten können in einer Iteration des Sicherheitsmikroprozesses mit Aspekten der Zugriffssicherheit angereichert werden.

Lokale Spezifikation der Sicherheit (SPA4)

Bei den Anforderungen zum Sicherheitsmikroprozess haben wir uns weiterhin für eine lokale Spezifikation der Sicherheit ausgesprochen, d.h., die Sicherheit wird nicht in seiner Gesamtheit

spezifiziert, sondern es werden kleine Einheiten, die funktionell modelliert werden, auch der Sicherheitsanalyse unterzogen. Dies hat zum Ziel, dass alle Systemteile einer Analyse der Zugriffssicherheit unterzogen und somit keine Bereiche übersehen werden. Das vorgestellte Vorgehen verfolgt diese Anforderung dahingehend, dass bei der Spezifikation der Geschäftsprozesse und der Anwendungsfälle sowie innerhalb der Analyse die Zugriffssicherheit lokal modelliert wird.

Betrachtung von Produktartefakten (SPA5)

Da unser Vorgehen an modernen objektorientierten Vorgehensweisen ausgerichtet ist, sind neben den eigentlichen Vorgängen auch die Produktartefakte, welche die Ein- und Ausgabeprodukte der Prozesse bilden, von besonderem Interesse. Wie bereits im Abschnitt zur sukzessiven Entwicklung der Sicherheit (SPA2) gezeigt wurde, arbeitet das im Abschnitt 5.3 beschriebene Vorgehen auf klassischen und sicherheitsspezifischen Produktartefakten. Eine Übersicht über die Produktartefakte der Anforderungsspezifikation zugriffssicherer Systeme ist im Abschnitt 5.4 zu finden.

Iterative Entwicklung (SPA6)

Insbesondere wegen der Komplexität heutiger Softwaresysteme unterstützen objektorientierte Entwicklungsprozesse eine iterative Vorgehensweise. Für die Entwicklung zugriffssicherer Systeme ist der Aspekt der frühzeitigen Identifikation und Reduktion der Risiken besonders tragend, denn Risiken im Bereich der Zugriffssicherheit sind bei der Entwicklung zu eliminieren. Durch die Einbettung der Anforderungsspezifikation zugriffssicherer Systeme können Iterationen über Phasen, die in einem objektorientierten Vorgehensmodell (zum Beispiel [Kru00, JBR99]) vorgegeben werden, angewendet werden (siehe hierzu Abbildung 5.5). Der eigentliche Prozess zur Modellierung der Zugriffssicherheit, der Sicherheitsmikroprozess aus Abschnitt 5.3.3, ist speziell für eine iterative Entwicklung ausgelegt worden und dieser wird zudem in allen Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme ausgeführt. Für die Modellierung zugriffssicherer Systeme gibt es noch eine dritte Iterationsmöglichkeit beim Entwurf und der Überprüfung von Maßnahmen gegen Bedrohungen innerhalb des Mikroprozesses (siehe Abbildung 5.9). Durch diese dreifache Iterationsfähigkeit wird durch das vorgestellte Vorgehen eine iterative Entwicklung in vollem Umfang unterstützt.

Sicherheitsmodellierung in der Geschäftsprozessanalyse (SPA7)

Eine Modellierung der Zugriffssicherheit in der Geschäftsprozessanalyse ist vor allem deshalb wichtig, da zu diesem Zeitpunkt Anwender und Anforderungsentwickler gemeinsam die Anforderungen entwickeln und dadurch die Aspekte der Zugriffssicherheit gemeinsam ermittelt und dokumentiert werden können. Beispielsweise führt die Tatsache, dass ein Mitarbeiter in einem Betrieb gewisse Unterlagen bei Verlassen des Büros in seinem Schreibtisch einsperrt, bei der Umsetzung in einen automatisierten und computergestützten Ablauf eine entscheidende Rolle. Diese Daten sich auch im System vor anderen Anwendern zu schützen. Der vorgestellte Prozess der Anforderungsspezifikation zugriffssicherer Systeme enthält einen Prozessabschnitt zur Geschäftsprozessmodellierung, um derartige Aspekte zu klären.

5.6 Zusammenfassung

Wie schon mehrfach erwähnt wurde, ist ein Ziel der Entwicklung zugriffssicherer Software die durchgängige und sukzessive Betrachtung von Aspekten der Zugriffssicherheit, insbesondere in der frühen Phase der Anforderungsspezifikation. Derzeit existieren einige Ansätze zum Security Engineering, die eine frühe Modellierung von Aspekten der Zugriffssicherheit unterstützen und diese durchgängig und sukzessive erarbeiten. Die Ermittlung der Anforderungen an zugriffssichere Systeme beginnt bei diesen Ansätzen entweder nach der Festlegung der funktionalen Anforderungen oder nach der Erstellung eines Pflichtenhefts (siehe Abschnitt 5.1). Sicherheit wird dabei zumeist ganzheitlich neben der Funktionalität modelliert, d.h., es wird eine eigenständige Aufteilung des Systems zur Ermittlung der Sicherheitsanforderungen durchgeführt. Eine Modellierung von Aspekten der Zugriffssicherheit bei der lokalen Modellierung der Funktionalität in kleinen Einheiten, wie beispielsweise einzelnen Geschäftsprozessen oder Anwendungsfällen, findet nicht statt. Die existierenden Ansätze beschreiben zumeist nur Vorgehensprozesse, gehen dabei auf Produktartefakte mit Ausnahme des vorgestellten CC-konformen Prozesses (vgl. Abschnitt 5.1.3) kaum ein. Für eine iterative Entwicklung werden bei [VM02] bereits erste Ideen vorgestellt, jedoch wurden diese in heutigen Vorgehensmethoden zur Entwicklung zugriffssicherer Software kaum umgesetzt.

Im Bereich der Standardentwicklung werden heute auch objektorientierte Vorgehensweisen eingesetzt, die bereits mit der frühen Phase der Geschäftsprozessmodellierung beginnen, kleine Portionen von Struktur und Verhalten in Anwendungsfällen ausarbeiten und starke Betonung auf Produktartefakte als Ein- und Ausgaben zu den Prozessen legen.

Der in diesem Kapitel vorgestellte Prozess zur Entwicklung einer Anforderungsspezifikation für zugriffssichere Systeme vereinigt die objektorientierte Standardentwicklung mit der Modellierung von zugriffssicheren Systemen und integriert so die durchgängige und sukzessive Entwicklung von sicheren Systemen in ein objektorientiertes Vorgehen. Prozessanforderungen, die aus gegenwärtigen Security Engineering Methoden und aus Standardentwicklungsmethoden abgeleitet worden sind, wurden in dem Prozess der Anforderungsspezifikation zugriffssicherer Systeme betrachtet. In diesem Kapitel wurde hierzu ein Prozess zur Anforderungsspezifikation zugriffssicherer Systeme, ein iterativ ausführbarer Sicherheitsmikroprozess, eine Integration der Anforderungsspezifikation in einen Softwarelebenszyklus sowie Produktartefakte für eine Modellierung zugriffssicherer Systeme eingeführt.

In den folgenden Kapiteln werden nun die Prozessabschnitte der Anforderungsspezifikation zugriffssicherer Systeme im Detail erklärt, d.h., es wird geklärt,

- welche klassischen Aktivitäten zur Modellierung des aktuellen Prozessabschnitts ohne Betrachtung der Aspekte der Zugriffssicherheit durchzuführen sind,
- welche Beschreibungstechniken dabei notwendig sind,
- welche zusätzlichen Aktivitäten für die Modellierung der Zugriffssicherheit notwendig sind,
- welche Beschreibungstechniken und Produktartefakte für eine Modellierung der Zugriffssicherheit zu erweitern bzw. neu hinzuzufügen sind (soweit dies nicht schon in diesem Kapitel geklärt wurde),

- wie die Abhängigkeiten der Produktartefakte für eine sukzessive Entwicklung aussehen, d.h., welche Artefakte in einem Arbeitsschritt aus anderen Artefakten erzeugt werden können (als Verfeinerung der Abhängigkeiten, wie sie im Abschnitt 5.4 eingeführt wurden),
- wie eine iterative Modellierung erfolgen kann und
- Beispiele, die die Aktivitäten, Beschreibungstechniken, Produktartefakte und iterative Entwicklung im Einsatz zeigen.

Für die Modellierung der Benutzerrechte wird in den folgenden Prozessabschnitten das in Kapitel 4 eingeführte Modell zur Beschreibung von Benutzerrechten angewendet.

6 Die Geschäftsprozessmodellierung zugriffssicherer Systeme

Die frühe Modellierung von Sicherheit haben wir bereits in den vorausgehenden Kapiteln motiviert. Im Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme wurde hierfür eine Aufteilung der Anforderungsdefinition in drei Prozessabschnitte vorgestellt. In diesem Kapitel gehen wir detailliert auf den ersten Prozessabschnitt der Anforderungsspezifikation zugriffssicherer Systeme ein, auf die Geschäftsprozessmodellierung zugriffssicherer Systeme.

Geschäftsprozesse sind allgemein betriebliche Prozesse, die zur Erstellung einer Unternehmensleistung beitragen. Die Geschäftsprozessmodellierung beschäftigt sich neben der eigentlichen Modellierung von Ablaufprozessen und deren Beziehungen untereinander auch mit der Modellierung von Merkmalen, wie beispielsweise Organisationseinheiten, Ein- und Ausgaben, Ressourcen, Informationen und Bedingungen. Bei der Modellierung sind sowohl fachliche Experten aus dem zu modellierenden Unternehmen bzw. der zu modellierenden Organisation vertreten als auch Anforderungsentwickler und Anwender. Gemeinsam werden die Unternehmens- bzw. Organisationsprozesse mit ihren Merkmalen erarbeitet und automatisierbare Teile identifiziert. Ein Überblick über die Geschäftsprozessmodellierung wird im Abschnitt 6.1 gegeben.

Bei der heutzutage üblichen Geschäftsprozessmodellierung werden jedoch Sicherheitsaspekte nicht betrachtet. Die Bearbeiter widmen sich der Funktionalität und den damit zusammenhängenden Merkmalen. Ergebnisse sind Ablauf-, Domänen- und Organisationsmodelle in Form von Klassen- und Aktivitätsdiagrammen.

Jedoch können im Rahmen der Geschäftsprozessmodellierung bereits eine Reihe von Sicherheitsaspekten ermittelt werden. Aus der manuellen Abarbeitung der Prozesse lassen sich bereits Schutzziele für Daten und Prozessaktivitäten ermitteln sowie Berechtigungshierarchien festlegen. Auf die Vertrauenswürdigkeit von Objekten kann beispielsweise dadurch geschlossen werden, dass Dokumente, welche die Objekte repräsentieren, im manuellen Ablauf nach Dienstende im Safe versperrt werden. Eine Eingangsbestätigung für ein Dokument stellt zum Beispiel einen Verbindlichkeitsnachweis dar. Derartig ermittelte Schutzziele in den Objekten und Abläufen der Geschäftsprozessmodellierung sind bei der Modellierung von Zugriffsrechten auf Daten und Ausführungsrechten von Aktivitäten heranzuziehen und müssen so nicht erst durch eine separate, detaillierte Analyse der Daten und Funktionen erarbeitet werden.

Zur Dokumentation der so erarbeiteten Schutzziele des Domänenmodells und der Prozesse sind geeignete Beschreibungstechniken notwendig. Im Abschnitt 6.2 stellen wir die Spezifikation von Schutzzielen in Struktur und Verhalten als Erweiterung von UML-Aktivitätsdiagrammen und UML-Klassendiagrammen vor. Die Bedeutung der Schutzziele wird im Kontext der Diagramme, gegebenenfalls formal, eingeführt.

Als Basis für eine Beschreibung der Zugriffs- und Ausführungsrechte sind die an den Prozessen beteiligten Akteure zu ermitteln. Hierfür ist im Rahmen der Geschäftsprozessmodellierung eine Modellierung der Organisationsstruktur, insbesondere der Bearbeiter, durchzuführen. Zu den Bearbeitern lassen sich die Akteure in Form von abstrakten Rollen und deren Hierarchie in Bezug auf die Zugriffs- und Ausführungsrechte bestimmen. Die Zugriffsrechte auf die Daten des Domänenmodells sowie Ausführungsrechte zu Geschäftsprozessen sowie die Hierarchisierung der Akteure wird im Abschnitt 6.3 betrachtet.

Abschnitt 6.4 stellt das Vorgehen bei der Geschäftsprozessmodellierung zugriffssicherer Systeme in Form eines Prozessmusters vor. Die Anwendung des Sicherheitsmikroprozesses aus Abschnitt 5.3.3, die im Rahmen der Geschäftsprozessmodellierung zugriffssicherer Systeme durchzuführen ist, wird exemplarisch für unsere *TimeTool*-Fallstudie gezeigt.

Abschließend betrachten wir im Abschnitt 6.5 die Produktartefakte der Geschäftsprozessmodellierung zugriffssicherer Systeme. Dabei gehen wir auf die Beziehung zu den Produktartefakten der allgemeinen Geschäftsprozessmodellierung näher ein.

6.1 Allgemeine Geschäftsprozessmodellierung

Bevor wir auf die Besonderheiten der Geschäftsprozessmodellierung zugriffssicherer Systeme eingehen, geben wir einen kurzen Überblick über Ziele, Aktivitäten und Produktartefakte der allgemeinen Geschäftsprozessmodellierung.

6.1.1 Motivation

Nach [EP00] starten viele UML-basierte Entwicklungsprozesse mit der Modellierung der Anwendungsfälle. Durch die Anwendungsfälle können zwar Aufgaben, die ein Akteur mithilfe des zu erstellenden Systems lösen will, beschrieben werden, aber es bleibt offen, inwiefern alle oder zumindest die wichtigsten Anwendungsfälle des Systems auf diese Art identifiziert werden. Weiterhin wird nicht geklärt, wie unterschiedliche Akteure in Prozessen zusammenarbeiten, welche Aktivitäten zu ihrem Aufgabenfeld gehören, welche Ziele sie mit ihren Arbeiten verfolgen und welche anderen Bearbeiter, Systeme oder Ressourcen dabei involviert sind.

Mit diesen Aspekten beschäftigt sich die Geschäftsprozessmodellierung. Die Abläufe und die Aufteilung der Unternehmung bzw. Organisation, die Ressourcen der Prozesse, die globalen Regeln, die Ziele sowie die Probleme werden innerhalb der Geschäftsprozessmodellierung untersucht. Informationen, die für das Unternehmen einen strategischen Wert besitzen, sind in einem *unvollständigen Domänenmodell* (siehe [Kru00, Rat00]) herauszuarbeiten.

Ein Geschäftsprozessmodell kann nach [EP00] niemals absolut genau und vollständig sein. Wenn zwei Betrachter ein und dasselbe System beschreiben, werden beide eine unterschiedliche Auffassung und Vorstellung vom System haben. Ziel der Geschäftsprozessmodellierung soll es auch nicht sein, jedes Detail vollständig bis ins kleinste Detail zu modellieren. Andernfalls würde das Modell nicht mehr fassbar und zu komplex werden. Das Geschäftsprozessmodell beschränkt sich somit auf die Kernkonzepte und Kernprozesse.

Unter diesen Voraussetzungen und Beschränkungen hat die Geschäftsprozessmodellierung folgende Aufgaben zu erfüllen [EP00]:

- Die entscheidenden Mechanismen einer Unternehmung bzw. Organisation müssen verstanden werden.
- Es ist eine Basis für die Entwicklung eines Systems zu schaffen. Beschreibungen der Geschäftsbereiche sind für die Identifikation der Systemunterstützung notwendig. Die Schlüsselanforderungen sind daraus abzuleiten.
- Die gegenwärtigen Geschäftsprozesse und die Geschäftsstruktur sind in Hinblick auf eine Automatisierung zu überarbeiten und gegebenenfalls zu verbessern.
- Die Struktur der verbesserten, erneuerten Unternehmung bzw. Organisation ist darzulegen.

In der Literatur hat sich bislang noch keine einheitliche Definition für den Begriff des *Geschäftsprozesses* herausgebildet (vgl. [Thu04, Wik]). Grundsätzlich kann zwischen zwei Interpretationen unterschieden werden. Die erste ist an die Definitionen von Geschäftsprozessen im Kontext des Business Process Reengineering angelehnt und sieht Geschäftsprozesse als Kernprozesse, die das Leistungsprogramm eines Unternehmens darstellen und als Ergebnis einen Wert für einen Kunden erzeugen. Der zweiten Interpretation liegt ein allgemeines Prozessverständnis zugrunde. Geschäftsprozesse werden als betriebliche Prozesse, die zur Erstellung der Unternehmensleistung beitragen, verstanden. Dazu gehören beispielsweise auch Prozesse der Produktentwicklung oder Marktforschung. Aus systemtheoretischer Sicht sind Geschäftsprozesse Folgen bestimmter diskreter Zustandsänderungen des betrachteten Systems Unternehmen.

Im Rahmen dieser Arbeit wird von dieser Unterscheidung abstrahiert und unter einem Geschäftsprozess die inhaltlich abgeschlossene, zeitlich-sachlogische Abfolge von Aktivitäten verstanden, die zur Bearbeitung eines für die Leistungserbringung des Unternehmens relevanten Objekts oder Dienstleistung erforderlich sind. Wir betrachten einen Geschäftsprozess als Muster für einen Arbeitsablauf [Thu04]. Dieses Muster gibt vor, aus welchen Aktivitäten sich der Prozess zusammensetzt, in welcher Reihenfolge und kausalen Ordnung die Aktivitäten ausgeführt werden müssen und welches Ziel damit erreicht wird. Die Daten, Informationen und Geschäftsobjekte, auf denen der Geschäftsprozess arbeitet sowie die Ergebnisse und Produkte, die im Geschäftsprozess erstellt werden, sind im Muster zu beschreiben. Im Rahmen der Betrachtung der Organisation des zu modellierenden Geschäftsbereichs interessieren wir uns insbesondere für die Akteure, welche die Geschäftsprozesse ausführen.

Weitere Anwendungsgebiete der Geschäftsprozessmodellierung sind Business Process Reengineering, Unterstützung bei der Auswahl und dem Customizing von betriebswirtschaftlicher Standardsoftware [Sta01], Requirements Engineering [Par98], Qualitätsmanagement [Bal96] (beispielsweise DIN ISO 9000 ff.) oder die Definition von Workflows sowie prozessorientiertes Wissensmanagement.

Geschäftsprozessmodell

Geschäftsprozessmodelle sind zweckorientierte, vereinfachte Abbildungen von Geschäftsprozessen. Aufgrund ihrer allgemeinen Modellcharakteristik dienen Geschäftsprozessmodelle der Dokumentation, Analyse und Gestaltung von Geschäftsprozessen sowie zur Unterstützung der Kommunikation über Geschäftsprozesse.

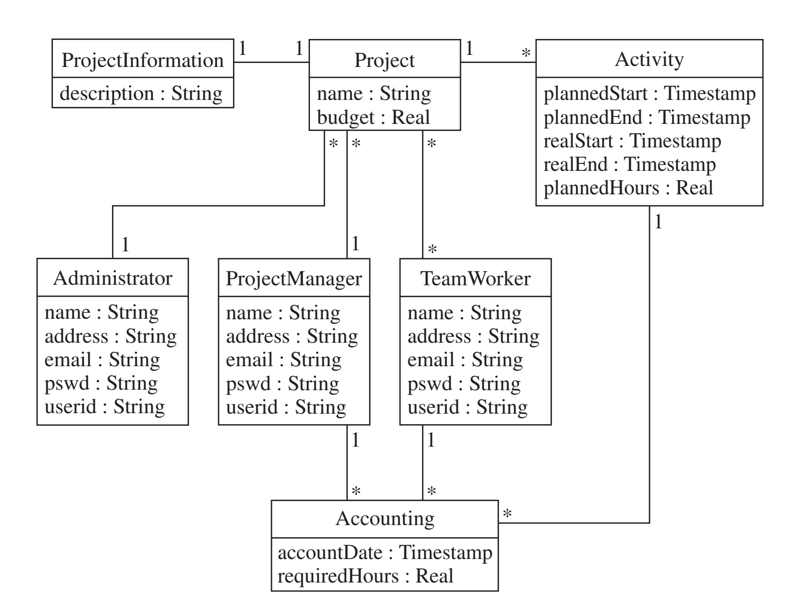


Abbildung 6.1: Domänenmodell der Geschäftsprozessmodellierung zur TimeTool-Fallstudie

6.1.2 Aktivitäten der Geschäftsprozessmodellierung

In der Geschäftsprozessmodellierung steht die Ermittlung der Kernprozesse und Kernkonzepte im Vordergrund. Bei der Erarbeitung dieser zentralen Prozesse und Konzepte sind nach [Kru00, DW98] die folgenden Aktivitäten auszuführen.

Zu Beginn ist die Organisation bzw. die Umgebung zu bewerten, in der das zu erstellende System eingebracht werden soll. Anforderungsentwickler, Auftraggeber, Anwender und Auftragnehmer müssen sich auf eine einheitliche Terminologie einigen.

Basierend auf der Bewertung der Organisation bzw. Unternehmung ist festzulegen, welche Geschäftsabläufe im System umgesetzt werden müssen. Anschließend wird festgelegt, wie die Systementwicklung fortgesetzt wird. Im Falle einer iterativen Entwicklung sind die Inhalte der einzelnen Iterationen zu definieren.

In Abhängigkeit von der geforderten Genauigkeit des Geschäftsmodells wird das Domänenmodell entwickelt. Ein Domänenmodell enthält die wichtigsten Typen von Objekten des zu beschreibenden Systems. Die Domänenobjekte repräsentieren die "Dinge", die im System existieren oder auftauchende Ereignisse (vgl. [JBR99]).

Falls keine tief greifenden Änderungen an den Abläufen der Organisation bzw. Unternehmung durchzuführen sind, können aus den Abläufen die Systemanforderungen abgeleitet werden. Bei der Entwicklung eines neuen Geschäftsbereichs müssen vorab die Abläufe erstmalig festgelegt werden.

6.1.3 Produktartefakte der Geschäftsprozessmodellierung

Die zentralen Produktartefakte der Geschäftsprozessmodellierung sind

- das *Domänenmodell*, ein Klassendiagramm, in dem die zentralen Typen der Objekte zur Beschreibung der Organisationsstruktur und statischer Konzepte modelliert werden und

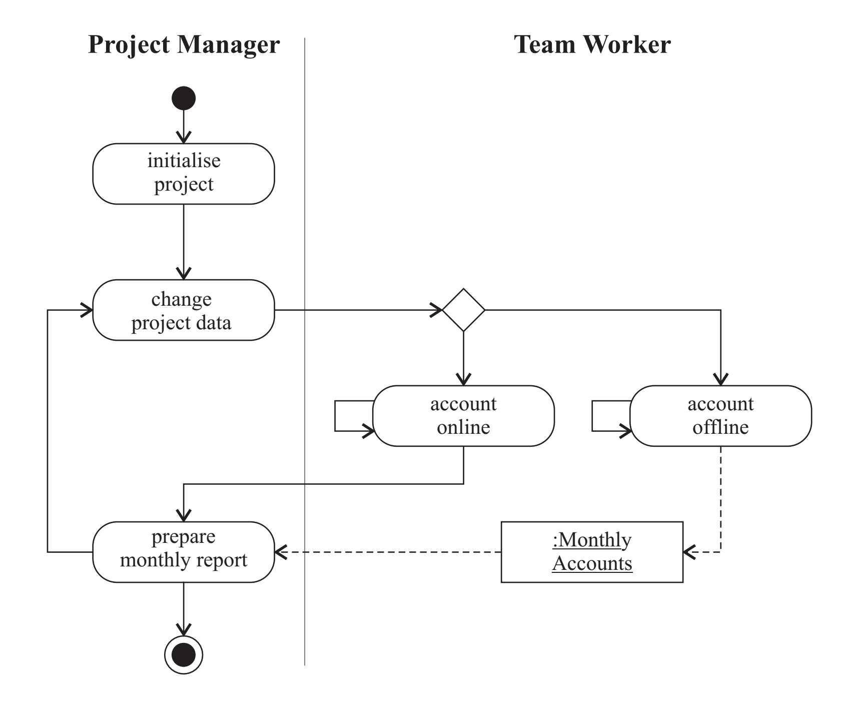


Abbildung 6.2: Ein Geschäftsprozess des TimeTool Projektverwaltungssystems

- das *Geschäftsprozessmodell*, Aktivitätsdiagramme, die die Abläufe der Unternehmung bzw. Organisation beschreiben.

Im *Geschäftsprozessmodell* sind neben den Abläufen als Aktivitätsfolgen noch eine Reihe weiterer Informationen beschrieben, die in Zusammenhang mit der Ablaufmodellierung zu erfassen sind:

- Die *Akteure*, eine Sammlung aller in den Abläufen involvierten Akteure. Zu den Akteuren gehören einzelne Personen, Gruppen, Organisationen, Unternehmen sowie Maschinen, die mit dem Geschäftsbereich interagieren.
- Die *Architektur* gibt einen Überblick über die Struktur und Zweck der Unternehmung bzw. Organisation. Sie dient als Kommunikationsbasis für Prozessanalysten und Projektmitarbeiter.
- In den *Ergänzungen* werden wichtige Informationen zum Geschäftsbereich sowie zum Kontext dokumentiert.
- Im *Glossar* werden Begrifflichkeiten und die vereinbarte Terminologie festgehalten.
- Die *Regeln* beschreiben globale Regeln und Beschränkungen, die bei der Entwicklung des Systems eingehalten werden müssen.

Abbildung 6.1 zeigt eine erste Version des Domänenmodells zu unserer *TimeTool*-Fallstudie. Hierbei wurden zentrale Geschäftsobjekte, (beispielsweise *Project*, *Activity* und *Accounting*) sowie Objekte der realen Welt (zum Beispiel *Administrator*, *Team Worker*) modelliert und in einem Klassendiagramm dargestellt.

Abbildung 6.2 zeigt einen exemplarischen Geschäftsprozess zur *TimeTool*-Fallstudie als Teil des Geschäftsprozessmodells.

6.2 Spezifikation der Schutzziele in Struktur und Verhalten

Wie wir in Kapitel 5 bereits erläutert und als Anforderung an den Prozess zur Entwicklung von sicheren Systemen definiert haben, ist es für eine durchgängige und integrierte Entwicklung notwendig, dass Sicherheit bereits bei der Erarbeitung der Funktionalität betrachtet wird. Damit die in den frühen Phasen erarbeiteten Anforderungen an die Sicherheit geeignet dokumentiert werden können, benötigen wir adäquate Beschreibungstechniken für die Annotation von Sicherheitsanforderungen im Kontext der grafischen Beschreibungssprache UML. In diesem Abschnitt führen wir derartige Beschreibungstechniken in Form von annotierten Schutzzielen zu den zentralen Produkten der Geschäftsprozessanalyse, dem Domänenmodell und den Geschäftsprozessen, ein.

Bei der Erarbeitung der Geschäftsprozesse und Geschäftsobjekte sind die Ziele festzuhalten, die bei einem Angriff verletzt werden können. Hierzu annotieren wir die potenziellen Angriffsziele auf der Granularität von Klassen und Aktivitäten mit Schutzzielen. Durch diese Annotation modellieren wir die Schutzbedürftigkeit der Objekte und Abläufe.

Bei der Analyse der potenziellen Angriffe beschränken wir uns auf die Verletzung der Schutzziele Vertraulichkeit, Integrität und Verbindlichkeit. Auf das Schutzziel Authentizität gehen wir im Rahmen der Geschäftsprozessmodellierung nur kurz ein, detailliert wird dieses Schutzziel innerhalb der Anwendungsfallmodellierung betrachtet (vgl. Kapitel 7). Auf die Schutzziele Verfügbarkeit und Anonymisierung gehen wir, wie bereits zu Beginn der Arbeit erläutert, nicht näher ein.

Mit Schutzzielen annotierte Diagramme stellen die Basis für die später durchzuführenden Modellierungen der Bedrohungen und Risiken dar. Jedes potenzielle Angriffsziel, welches als solches erkannt und im Diagramm annotiert wird, ist nach Bedrohungen zu untersuchen und zu den Bedrohungen sind die Risiken zu abzuschätzen. Wenn das Risiko einer Bedrohung unter Betrachtung der System- und Umgebungsanforderungen nicht verneint oder nicht als geringfügig eingestuft werden kann, müssen geeignete Maßnahmen gesucht werden und die Maßnahmen sind bezüglich ihrer Tauglichkeit zu überprüfen.

Wie bereits erwähnt, verwenden wir die grafische Beschreibungstechnik UML einerseits deshalb, weil sie sich innerhalb von objektorientierten Vorgehensmodellen zum Modellierungsstandard entwickelt hat. Ein weiterer entscheidender Vorteil, der für Einsatz der UML propagiert, ist die Erweiterbarkeit der UML. Durch ihre Erweiterungsfähigkeit in Form von *Stereotypen*, *Tagged Values* und *Constraints* (vgl. [BRJ98, EP00]) ermöglicht sie es, die notwendigen Schutzzielannotationen in den hier verwendeten Klassen- und Aktivitätsdiagrammen einzubringen.

Im Folgenden stellen wir die notwendigen Erweiterungen der UML für die Spezifikation von Schutzzielen im Domänenmodell und in den Abläufen der Geschäftsprozessmodellierung vor. Bei der Ablaufmodellierung zeigen wir auf zwei unterschiedliche Detailstufen die Schutzzielannotation. Es können entweder nur die reinen Aktivitätsfolgen betrachtet werden oder in detaillierter Form die zwischen den Aktivitäten ausgetauschten Flussobjekte ebenfalls annotiert werden. Die Spezifikation der Schutzziele an Flussobjekten kann als Erweiterung der Aktivitätsfolgen betrachtet werden und ist dann durchzuführen, falls die Geschäftsprozessmodellierung die Modellierung auf dieser detaillierten Ebene erfordert.

6.2.1 Schutzziele in den Geschäftsobjekten des Domänenmodells

Im Rahmen der Identifikation von Geschäftsobjekten innerhalb der Geschäftsprozessmodellierung sind die Geschäftsobjekte bezüglich ihrer potenziellen Angriffsmöglichkeiten zu untersuchen. Als zentrale Fragestellung für die Untersuchung der Angriffe ist zu klären, inwiefern einzelne oder mehrere Attribute der Geschäftsobjekte potenzielles Angriffsziel für

- unerlaubten lesendem Zugriff,
- unerlaubter Manipulation oder
- Verbindlichkeit des lesenden Zugriffs bzw. der Manipulation

sind. Falls ein oder mehrere dieser potenziellen Angriffsziele für ein Geschäftsobjekt erfüllt sind, so ist die zugehörige Klasse im Domänenmodell der Geschäftsprozessanalyse zu annotieren.

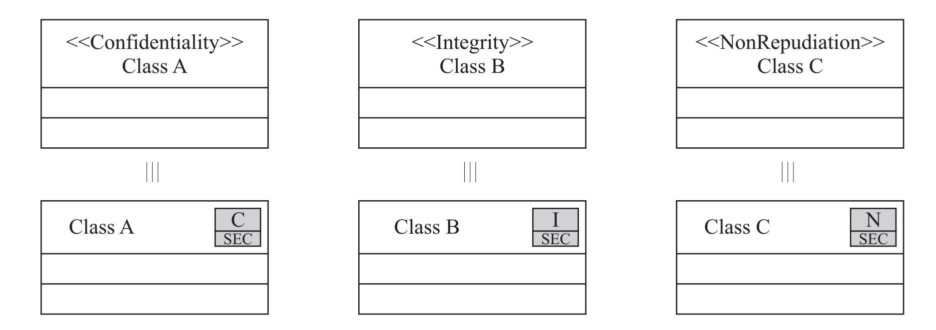


Abbildung 6.3: Stereotypen zu den Schutzzielen der Geschäftsobjekte

Zur Annotation von potenziellen Angriffen führen die in Abbildung 6.3 gezeigten Stereotypen ein. Wir kennzeichnen ein Klasse mit dem Stereotyp ≪*Confidentiality*≫, falls die Klasse Attribute enthält, die vor unerlaubten Leseangriffen zu schützen sind. Sind Attribute der Klasse vor unerlaubter Veränderung zu schützen, so annotieren wir die Klasse mit dem Stereotyp ≪*Integrity*≫. Falls der Zugriff auf Attribute einer Klasse nicht abstreitbar sein muss, so kennzeichnen wir die Klasse mit dem Schutzziel ≪*NonRepudiation*≫.

Zur Abgrenzung von allgemeinen Stereotypen und den eingeführten Stereotypen für die Schutzziele führen wir eigene Schutzziel-Icons ein. In Abbildung 6.3 sind die drei Schutzziel-Icons zu den eingeführten Stereotypen abgebildet. Die Schutzziel-Icons enthalten das Schlüsselwort SEC (für Security) und den ersten Buchstaben des zu schützenden Schutzziels.

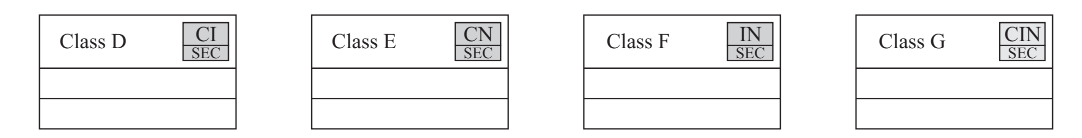


Abbildung 6.4: Kombinierte Stereotypen zu den Schutzzielen der Geschäftsobjekte

Attribute sind nicht nur gegen einzelne Angriffe zu schützen, sehr häufig sind Kombinationen möglich. So kann beispielsweise ein unerlaubtes Lesen und ein unerlaubtes Manipulieren eines Attributes eines Objektes ein potenzieller Angriff sein. Um derartige kombinierte Angriffe in einem einzigen Klassendiagramm zu annotieren, benötigen wir komplexe Stereotypen, sodass

wir derartige Schutzziele im Klassendiagramm ausdrücken zu können. Da jede Kombinationsmöglichkeit der vorgestellten einzelnen Schutzziele möglich ist, bilden wir die Potenzmenge der in Abbildung 6.3 definierten Stereotypen. Abbildung 6.4 zeigt die möglichen kombinierten Stereotypen als Schutzziel-Icons. Zur Vollständigkeit der Potenzmenge über den gegebenen Stereotypen zu den Schutzzielen fehlt noch das *Element für die leere Menge*, das sich durch eine Klasse ohne Schutzziel-Stereotyp ausdrücken lässt.

Kombinierte Stereotypen bilden eine *AND*-Verknüpfung zwischen den einzelnen Stereotypen. Wird beispielsweise ein Objekt mit den Schutzzielen *Integrität* und *Vertraulichkeit* gekennzeichnet, so ist das gekennzeichnete Objekt sowohl potenzielles Angriffsziel für unerlaubtes Lesen als auch für unerlaubte Manipulation.

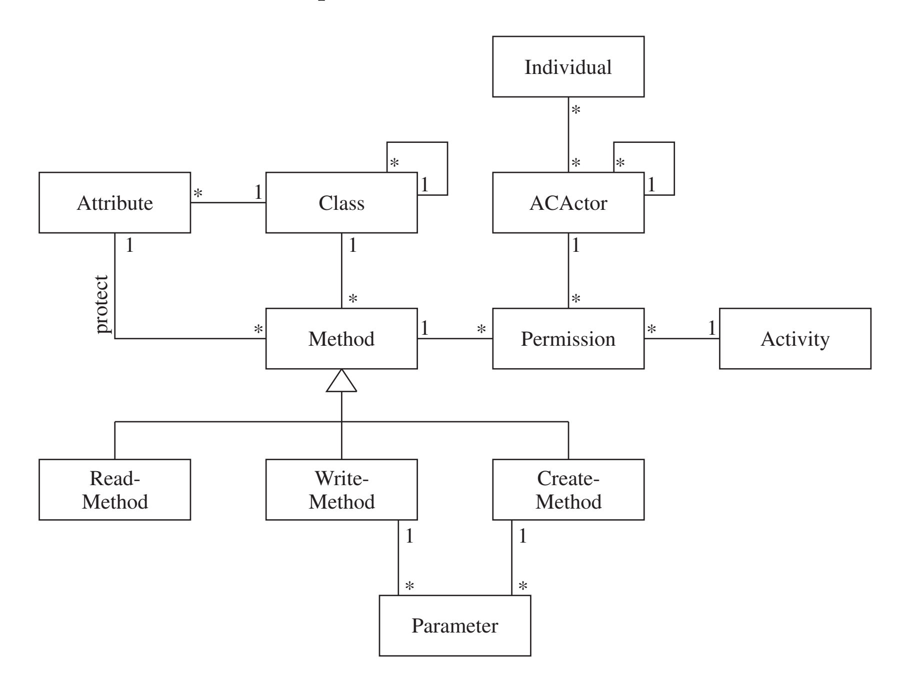


Abbildung 6.5: Metamodell zur Spezifikation von Berechtigungen

Im Folgenden betrachten wir die genaue Bedeutung der in Abbildung 6.3 vorgestellten Stereotypen zu den Schutzzielen Vertraulichkeit, Integrität und Nichtabstreitbarkeit in Klassendiagrammen. Basis für die formale Betrachtung der Schutzziele im Kontext der Geschäftsobjekte und Aktivitäten bilden der Spezifikationsrahmen P-MOS aus Abschnitt 4.3, die formale Modellierung von Benutzerrechten aus Abschnitt 4.4 sowie das in Abbildung 6.5 dargestellte Metamodell zur Spezifikation von Berechtigungen. Das Berechtigungsmodell basiert auf dem Spezifikationsrahmen P-MOS.

6.2.1.1 Schutzziel Vertraulichkeit im Klassendiagramm

Seien $A_1 : T_1, \ldots, A_n : T_n$ Attribute der Klasse C (T_ν ist entweder Klassenname oder ein Basistyp). Falls die Klasse C mit dem Schutzziel-Stereotyp ≪*Confidentiality*≫ annotiert wird, gibt es eine Teilmenge $A_{i_1} : T_{i_1}, \ldots, A_{i_m} : T_{i_m}$ der Attribute der Klasse C, deren *lesender Zugriff* eingeschränkt ist.

Für den lesenden Zugriff führen wir Lese-Zugriffsmethoden

$$\begin{aligned}&\boldsymbol{funct}\; read_{C,A_{i_1}}(ACActor) : T_{i_1},\\&\ldots\\&\boldsymbol{funct}\; read_{C,A_{i_m}}(ACActor) : T_{i_m}\end{aligned}$$

ein, die einen lesenden Zugriff auf ein Attribut A_μ für einen Akteur genau dann erlauben, wenn die zugehörige Leseberechtigung $perm_{C,read_{C,A_\mu}}(ACActor, C)$ für den Akteur erfüllt ist.

Für die Zuordnung von Lese-Zugriffsmethoden und Attributen, deren lesender Zugriff eingeschränkt ist, führen wir die Relation

$$protect \subseteq \mathcal{A} \times \mathcal{MI}$$

ein, wobei $\mathcal{A}$ die Menge aller Attribute und $\mathcal{MI}$ die Menge aller Methodenidentifikatoren sei. Ein Attribut A und ein Methodenidentifikator sind ein Element der Relation *protect*,

$$(A, read_{C,A}) \in protect,$$

wenn die Lese-Zugriffsmethode $read_{C,A}$ der Klasse C den Lesezugriff auf das Attribut A einschränkt.

Unter Betrachtung der eingeführten Relation *protect* und der vorgestellten *Lese-Zugriffsmethoden* mit Leseberechtigung können wir nun das Schutzziel-Stereotyp ≪*Confidentiality*≫ folgendermaßen beschreiben:

$$\begin{aligned}&\forall A \in C\ .\ \forall a : ACActor :\\&\exists\, read_{C,\,A_\mu} : (A_\mu,\ read_{C,\,A_\mu}) \in protect\ \Rightarrow\\&\quad \exists\, perm_{C,\,read_{C,\,A_\mu}} :\\&\qquad perm_{C,\,read_{C,\,A_\mu}}(a, C) = true\ \Leftrightarrow\ read_{C,A_\mu}(a) = \overline{A}_\mu\ \vee\\&\qquad perm_{C,\,read_{C,\,A_\mu}}(a, C) = false\ \Leftrightarrow\ read_{C,A_\mu}(a) = \bot\end{aligned}$$

Alle Attribute der Klasse C, für die spezielle Lese-Zugriffsmethoden existieren, können genau dann von Akteuren gelesen werden, wenn die zugehörige Leseberechtigung für den ausführenden Akteur *true* liefert. Als Ergebnis liefert die Lese-Zugriffsmethode den Wert des Attributs A_μ. Zur Unterscheidung zwischen Namen und Wert des Attributs kennzeichnen wir den Wert (die Belegung) des Attributs A durch $\overline{A}$.

Ist die zugehörige Leseberechtigung nicht erfüllt, so liefert die Lese-Zugriffsmethode das spezielle Element $\bot$ ("Bottom") als Abstraktion einer Fehlermeldung oder Exception zurück.

Beispiel
Abbildung 6.6 zeigt einen Ausschnitt der Geschäftsobjekte der *TimeTool*-Fallstudie. In den Klassen *TeamWorker* und *Accounting* enthalten jeweils die grau hinterlegten Attribute vertrauliche Information. Bei der *Klasse TeamWorker* sind Name und E-Mail-Adresse ohne Einschränkung lesbar, hingegen sind die Adresse sowie Benutzerkennung und Passwort vor unbefugtem lesenden Zugriff zu schützen. Die zu schützenden Attribute sind hier eine *echte Teilmenge* der Klassenattribute. In der Klasse *Accounting* enthalten alle Klassenattribute vertrauliche Information, d.h., die zu schützenden Attribute bilden eine *unechte Teilmenge* der Klassenattribute.

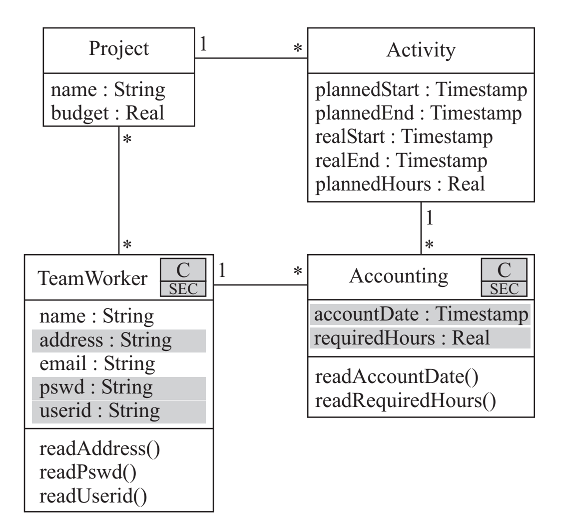


Abbildung 6.6: Schutzziel Vertraulichkeit in TimeTool-Geschäftsobjekten

6.2.1.2 Schutzziel Integrität im Klassendiagramm

Seien $A_1 : T_1, \ldots, A_n : T_n$ Attribute der Klasse C (T_ν ist entweder Klassenname oder ein Basistyp). Falls die Klasse C mit dem Schutzziel-Stereotyp ≪*Integrity*≫ annotiert wird, gibt es eine Teilmenge $A_{i_1} : T_{i_1}, \ldots, A_{i_m} : T_{i_m}$ der Attribute der Klasse C, deren *schreibender* oder *erzeugender Zugriff* eingeschränkt ist.

Für den schreibenden Zugriff führen wir Schreib-Zugriffsmethoden

$$\textbf{\textit{funct}}\ write_{C,A_{i_1}}(a : ACActor, o_{i_1} : T_{i_1})$$
$$\ldots$$
$$\textbf{\textit{funct}}\ write_{C,X_{i_m}}(a : ACActor, o_{i_m} : T_{i_m})$$

ein, die einen schreibenden Zugriff auf ein Attribut A_μ genau dann erlauben, wenn die zugehörige Schreibberechtigung $perm_{C,write_{C,A_\mu}}(ACActor, C, T)$ für den Akteur erfüllt ist.

Wir erweitern die Relation *protect* für die Zuordnung von Schreib-Zugriffsmethoden auf Attribute. Wir fügen das Tupel aus Attribut A und Methodenidentifikator genau dann in unsere Relation *protect* aus Abschnitt 6.2.1.1 ein

$$(A, write_{C,A}) \in protect,$$

wenn die Schreib-Zugriffsmethode $write_{C,A}$ der Klasse C den Schreibzugriff auf das Attribut A einschränkt.

Für den erzeugenden Zugriff führen wir Create-Zugriffsmethoden

$$\textbf{\textit{funct}}\ create_{C,A_{i_1}}(a : ACActor, o_{i_{1_1}} : T_{i_{1_1}}, \ldots, o_{i_{1_p}} : T_{i_{1_p}})$$
$$\ldots$$
$$\textbf{\textit{funct}}\ create_{C,A_{i_m}}(a : ACActor, o_{i_{m_1}} : T_{i_{m_1}}, \ldots, o_{i_{m_p}} : T_{i_{m_p}})$$

ein, die einen erzeugenden Zugriff auf ein Attribut A_μ vom Typ T genau dann erlauben, wenn die zugehörige Create-Berechtigung $perm_{C,create_{C,A_\mu}}(ACActor, C, T_{\mu 1}, \ldots, T_{\mu p})$ für den Akteur erfüllt ist.

Wir fügen das Tupel aus Attribut A und Methodenidentifikator der Create-Methode genau dann in unsere Relation *protect* ein

$$(A, create_{C,A}) \in protect,$$

wenn die Create-Zugriffsmethode $create_{C,A}$ der Klasse C den erzeugenden Zugriff auf das Attribut A einschränkt.

Unter Betrachtung der Relation *protect* und der vorgestellten *Schreib-* und *Create-Zugriffsmethoden* mit Berechtigungen können wir nun das Schutzziel-Stereotyp ≪*Integrity*≫ folgendermaßen beschreiben:

$$\begin{aligned}
&\forall\, A \in C \;\; . \;\forall\, a : ACActor : \\
&[\, \exists\, write_{C,\,A_\mu} : (A_\mu,\, write_{C,\,A_\mu}) \in protect \;\Rightarrow \\
&\quad \exists\, perm_{C,\,write_{C,\,A_\mu}} : \\
&\qquad perm_{C,\,write_{C,\,A_\mu}}(a, C, o_\mu) = true \;\Leftrightarrow\; \overline{A}_\mu = o_\mu \\
&\qquad perm_{C,\,write_{C,\,A_\mu}}(a, C, o_\mu) = false \;\Leftrightarrow\; write_{C,A_\mu}(a, o_\mu) = \bot \,] \;\wedge \\
&[\, \exists\, create_{C,\,A_\mu} : (A_\mu,\, create_{C,\,A_\mu}) \in protect \;\Rightarrow \\
&\quad \exists\, perm_{C,\,create_{C,\,A_\mu}} : \\
&\qquad perm_{C,\,create_{C,\,A_\mu}}(a, C, o_{\mu 1}, \ldots, o_{\mu p}) = true \;\Leftrightarrow\; \overline{A}_\mu = T(o_{\mu 1}, \ldots, o_{\mu p}) \;\vee \\
&\qquad perm_{C,\,create_{C,\,A_\mu}}(a, C, o_{\mu 1}, \ldots, o_{\mu p}) = false \;\Leftrightarrow\; create_{C,\,A_\mu}(a, o_1, \ldots, o_p) = \bot \,]
\end{aligned}$$

Alle Attribute der Klasse C, für die Schreib- oder Create-Zugriffe existieren, können genau dann von den Akteuren geschrieben oder erzeugt werden, wenn die zugehörige Schreib- oder Create-Berechtigung für den Akteur erfüllt ist.

Die Schreib-Zugriffsmethode weist dem Wert des Attributs A_μ den Wert des Parameters o_μ genau dann zu, wenn die Schreibberechtigung auf das Attribut A_μ in der Schreib-Zugriffsmethode für den Akteur a erfüllt ist. Die Create-Zugriffsmethode erzeugt ein neues Objekt vom Typ T_μ des Attributs A_μ und weist die Objekt-ID des neuen Objekts dem Wert des Attributs A_μ zu, wenn die Create-Berechtigung für das Attribut A_μ beim Ausführen der zugehörigen Create-Zugriffsmethode für den Akteur a erfüllt ist. Die Parameter $o_{\mu 1}, \ldots, o_{\mu p}$ der Create-Zugriffsmethode werden zur Initialisierung des Objekts T_μ benötigt.

Ist bei der Ausführung einer Schreib- oder Create-Zugriffsmethode die Berechtigung nicht erfüllt, so liefert die Schreib- bzw. Create-Zugriffsmethode das spezielle Element $\bot$ ("Bottom") als Abstraktion einer Fehlermeldung oder Exception zurück.

Beispiel
In Abbildung 6.7 sind einige Geschäftsobjekte der *TimeTool*-Fallstudie mit dem Stereotyp ≪*Integrity*≫ annotiert. In allen vier Geschäftsobjekten sind jeweils alle Attribute vor unerlaubten Veränderungen zu schützen, wie durch die grau hinterlegten Attribute gekennzeichnet wurde. In den Klassen *Project* und *ProjectManager* existieren jeweils Create-Zugriffsmethoden, deren Ausführung ebenfalls durch Berechtigungen einzuschränken ist.

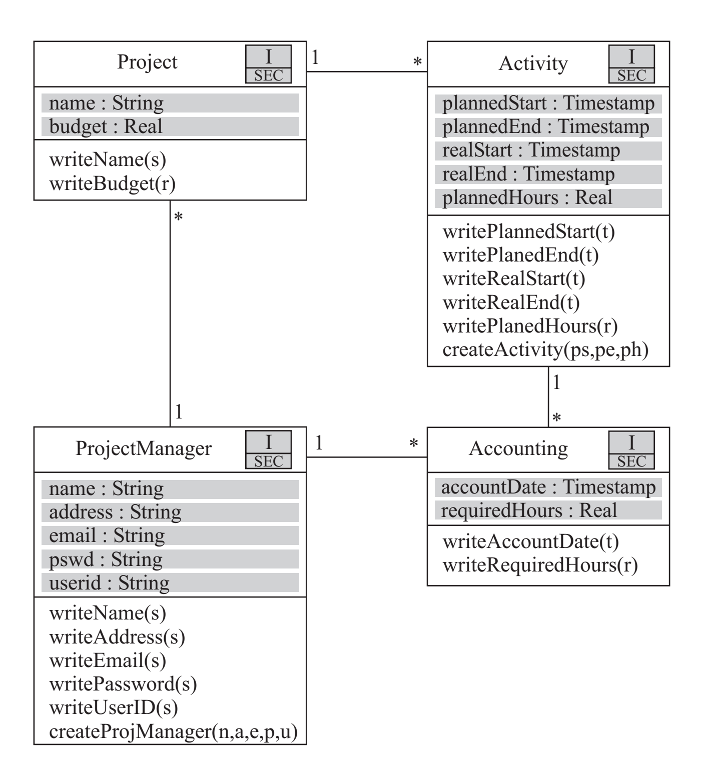


Abbildung 6.7: Schutzziel Integrität in TimeTool-Geschäftsobjekten

6.2.1.3 Schutzziel Verbindlichkeit im Klassendiagramm

Seien $A_1 : T_1, \ldots, A_n : T_n$ Attribute der Klasse C (T_ν ist entweder Klassenname oder ein Basistyp). Falls die Klasse C mit dem Schutzziel-Stereotyp ≪*NonRepudiation*≫ annotiert wird, gibt es eine Teilmenge $A_{i_1} : T_{i_1}, \ldots, A_{i_m} : T_{i_m}$ der Attribute der Klasse C, deren lesender, schreibender oder erzeugender Zugriff durch einen Akteur a im Nachhinein nicht abgestritten werden kann.

Für den verbindlichen Zugriff auf ein einzelnes Attribut führen wir für den Methodenzugriff eine Zugriffshistorie ein, d.h., für Attribute, deren Zugriff im Nachhinein nicht abgestritten werden kann, wird der Zugriff in einer Zugriffshistorie gespeichert. Zur Gewährleistung der Nichtabstreitbarkeit führen wir deshalb eine Relation

$$history \subseteq \mathcal{AC} \times \mathcal{OB} \times \mathcal{MI} \times [\mathcal{P}]$$

ein, wobei $\mathcal{AC}$ die Menge aller Akteure, $\mathcal{OB}$ die Menge aller Objekte, $\mathcal{MI}$ die Menge aller Methodenidentifikatoren und $[\mathcal{P}]$ eine Sequenz über Parametern sei.

Für Attribute, deren Zugriff verbindlich ist, gilt: Ein Akteur A, ein Objekt O als Attributbelegung einer Klasse C, ein Methodenidentifikator $methID_{C,x}$ und eine Sequenz von Parametern $[p_1, \ldots p_p]$ sind genau dann ein Element der Relation $history$,

$$(A, O, methID_{C,x}, [p_1, \ldots p_p]) \in history,$$

wenn der Akteur mittels einer Zugriffsmethode $methID_{C,x}$ mit Parameterbelegung $p_1, \ldots, p_p$ auf ein Attribut eines Objekt der Klasse C mit der Belegung O zugegriffen hat. Voraussetzung

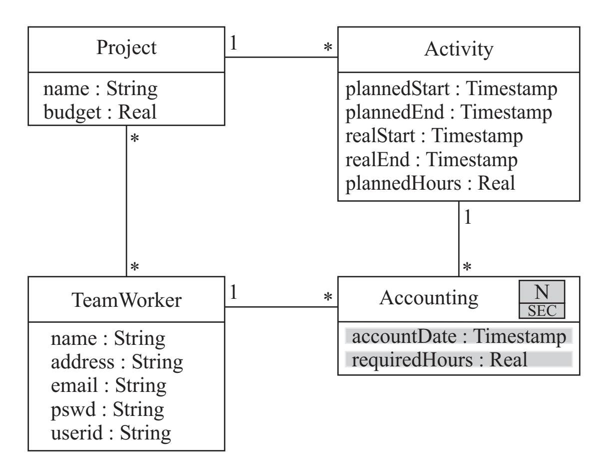


Abbildung 6.8: Schutzziel Verbindlichkeit in TimeTool-Geschäftsobjekten

für einen Lese-, Schreib- oder Create-Zugriff ist eine gültige Zugriffsberechtigung

$$perm_{methID_{C,x}}(a, C, p_1, \ldots, p_n) = true,$$

andernfalls erfolgt kein Zugriff und es wird auch kein Historienelement der obigen Form abgelegt.

Für alle Zugriffsmethoden, die auf ein Attribut zugreifen, dessen Zugriff verbindlich sein muss, ändern wir die Zugriffsfunktionen dahingehend ab, dass nach berechtigter Ausführung des Methodenzugriffs ein derartiges Historienelement geschrieben wird. Wir nehmen dazu an, dass es sich bei der erweiterten Methode um eine unteilbare Methode handelt. Somit kann kein Fall eintreten, bei dem ein verbindlicher Zugriff ohne Erzeugung eines Historienelements oder die Erzeugung eines Historienelements ohne verbindlichen Zugriff stattfindet.

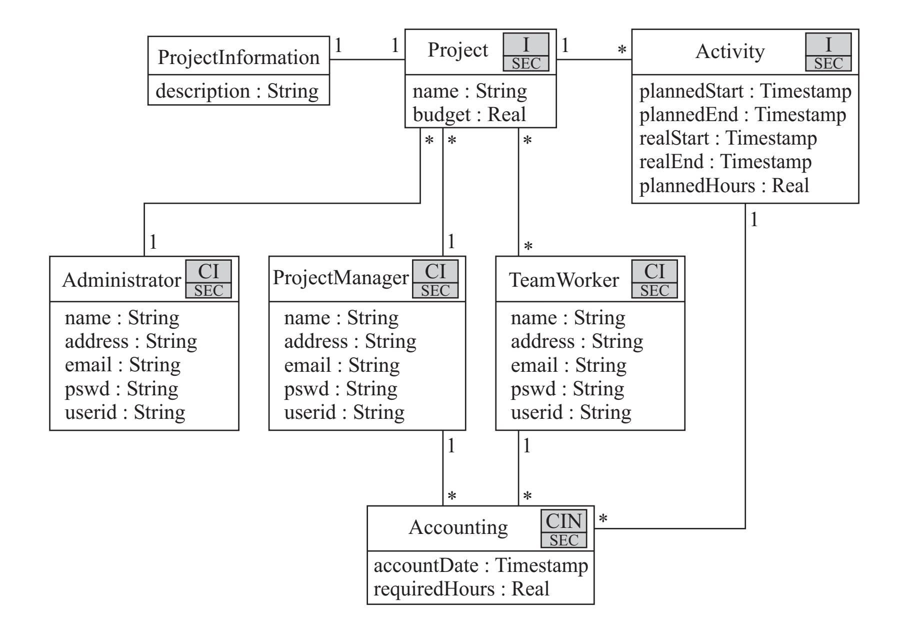


Abbildung 6.9: Kombinierte Schutzziel-Stereotypen in TimeTool-Geschäftsobjekten

Beispiel
Ein Ausschnitt der Geschäftsobjekte der *TimeTool*-Fallstudie ist in Abbildung 6.8 dargestellt. Die Klasse *Activity* ist hierbei mit dem Stereotyp *Verbindlichkeit* annotiert, da die grau hinterlegten Attribute *accountDate* und *requiredHours* verbindliche Attribute sind. Jeder Zugriff auf diese beiden Attribute muss entsprechend protokolliert werden.

Die Abbildungen 6.6, 6.7 und 6.8 zeigen Ausschnitte aus den Schutzziel-Stereotypen Vertraulichkeit, Integrität und Verbindlichkeit in den *TimeTool*-Geschäftsobjekten. Wie in Abbildung 6.4 dargestellt wurde, können Schutzziel-Stereotypen kombiniert werden, sodass nicht jedes einzelne Schutzziel in einem eigenen Diagramm modelliert werden muss. Abbildung 6.9 zeigt die kombinierten Schutzzielannotationen zu den *TimeTool*-Geschäftsobjekten.

6.2.2 Schutzziele in Abläufen der Geschäftsprozessmodellierung

Neben den im Abschnitt 6.2.1 vorgestellten Schutzzielen in Geschäftsobjekten des Domänenmodells sind Schutzziele auch in den Geschäftsprozessen zu identifizieren. Innerhalb der Aktivitäten von Prozessabläufen werden Daten verarbeitet, sodass die Daten ebenfalls vor unerlaubt lesendem Zugriff und vor unerlaubter Manipulation zu schützen sind. Auch die Ausführung einer Aktivität muss bei kritischen Geschäftsprozessen zu einem späteren Zeitpunkt nachvollzogen werden können. Wird im Rahmen der Geschäftsprozessmodellierung festgestellt, dass Aktivitäten derartigen potenziellen Angriffen ausgesetzt sind, annotieren wir Aktivitäten mit den in Abbildung 6.10 eingeführten Schutzzielen.

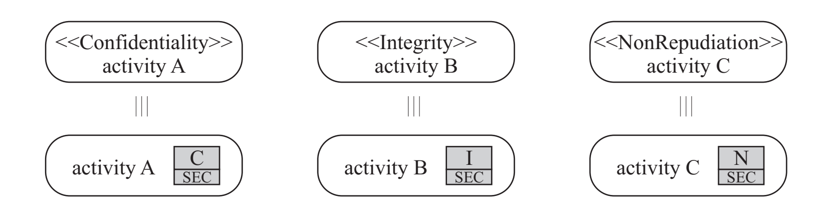


Abbildung 6.10: Schutzziele in Aktivitätsdiagrammen

Wir verwenden analog zu den eingeführten Stereotypen für Klassendiagramme die Schutzziel-Icons ≪*Confidentiality*≫, ≪*Integrity*≫ und ≪*NonRepudiation*≫ innerhalb von Aktivitäten eines Aktivitätsdiagramms. Da auch Aktivitäten häufig nicht nur einzelnen Angriffen ausgesetzt sind, erlauben wir auch eine Kombination der Schutzziel-Icons, wie dies in Abbildung 6.11 dargestellt ist.

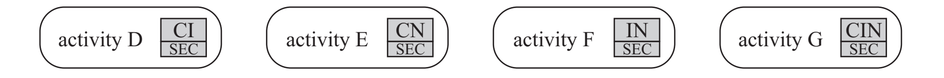


Abbildung 6.11: Kombinationen von Schutzzielen in Aktivitätsdiagrammen

Im Folgenden betrachten wir die genaue Bedeutung der vorgestellten Schutzziele in Aktivitätsdiagrammen. Vorab gehen wir noch kurz auf den Zusammenhang zwischen Aktivitäten und Methoden ein.

6.2.2.1 Zusammenhang zwischen Aktivitäten und Methoden

Zur Festlegung der Schutzziele in Aktivitätsdiagrammen müssen wir vorab festlegen, wie Aktivitäten eines Aktivitätsdiagramms mit Methoden zusammenhängen. Dabei gehen wir davon aus, dass die im Rahmen der Geschäftsprozessmodellierung betrachteten Aktivitätsdiagramme derart verfeinert werden können, dass in jeder Aktivität genau eine Methode aufgerufen wird.

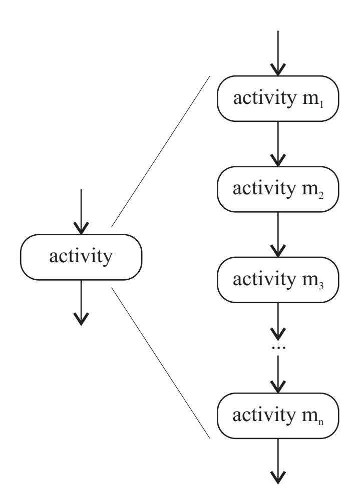


Abbildung 6.12: Verfeinerung der Aktivitätsdiagramme zu den Geschäftsprozessen

In den derart verfeinerten Aktivitätsdiagrammen unterscheiden wir zwei verschiedene Typen von Aktivitäten:

1. Der Aktivität kann eine Datenzugriffsmethode oder ein Systemaufruf zugeordnet werden. Mithilfe der zugeordneten Methode können nur Daten gelesen, geschrieben oder erzeugt werden. Die Aktivität wird als unteilbare Aktion betrachtet.
2. Der Aktivität kann eine Vorgehensmethode zugeordnet werden. Mit Hilfe dieser zugeordneten Methode wird ein weiterer Ablauf festgelegt, der Datenzugriffe und Vorgehensabläufe regelt. Die Aktivität ist in sich weiter teilbar.

6.2.2.2 Schutzziele in Aktivitätsdiagrammen

Betrachten wir nun die Schutzziele zu einer Aktivität, so wird im Fall 1 (Aktivität mit zugeordneter Datenzugriffsmethode) Folgendes festgelegt: Werden in der zur Aktivität zugeordneten Methode vertrauliche Daten gelesen, manipuliert oder muss ein Zugriff nachgewiesen werden, so wird die Aktivität mit dem Schutzziel ≪*Confidentiality*≫, ≪*Integrity*≫ oder ≪*NonRepudiation*≫ gekennzeichnet. Vertrauliche, integrielle oder verbindliche Daten sind dadurch gekennzeichnet, dass die zugehörigen Klassen mit den Schutzzielen ≪*Confidentiality*≫, ≪*Integrity*≫ und ≪*NonRepudiation*≫ annotiert sind (siehe Abschnitt 6.2.1).

Im 2. Fall einer zugeordneten Vorgehensmethode kann der Ablauf weiter "aufgebrochen" werden, sodass der Ablauf wiederum eine Folge von Aktivitäten mit Datenzugriffen und Vorgehensabläufen regelt.

Setzt sich nun ein Ablauf einer Aktivität aus einer Menge von Aktivitäten mit zugeordneten Methodenaufrufen zusammen, so ergibt sich das Schutzziel für die übergeordnete Aktivität als *logisches UND* der Schutzziele der Teilaktivitäten. Werden beispielsweise in einer Methode zu einer Teilaktivität vertrauliche Daten gelesen und müssen die Zugriffe in einer Methode zu einer weiteren Teilaktivität verbindlich sein, so werden an der übergeordneten Aktivität die Schutzziele ≪*Confidentiality*≫ und ≪*NonRepudiation*≫ annotiert.

Für die Festlegung der Schutzziele einer Aktivität sind somit immer die Datenzugriffe und Systemaufrufe innerhalb des verfeinerten Ablaufs ausschlaggebend. Enthält der verfeinerte Ablauf eine Aktivität mit zugeordneter Vorgehensmethode, so werden die Schutzziele wiederum durch die Verfeinerung der Vorgehensmethode ermittelt. Diese Verfeinerung wird so lange fortgesetzt, bis der verfeinerte Ablauf nur noch aus Datenzugriffen und Systemaufrufen besteht. Anschließend kann das Schutzziel der übergeordneten Aktivität berechnet und rekursiv zurückgesetzt werden. Die Schutzziele einer Aktivität errechnen sich somit aus der transitiven Hülle der Datenzugriffe in den zugeordneten Methoden der Teilaktivitäten. Auf die Bildung der transitiven Hülle gehen wir in Zusammenhang mit der Ermittlung von Benutzerrechten in Abschnitt 7.4.2 näher ein.

Sind in den Abläufen Verzweigungen (in Form von Bedingungen oder parallelen Abläufen) enthalten, so werden für die Ermittlung der Schutzziele stets alle Wege herangezogen.

Nach dem derartigen Verfahren können die Schutzziele zu Aktivitäten genau dann berechnet werden, wenn die Teilabläufe festgelegt und die Schutzziele zu den Datenobjekten und Systemfunktionen festgelegt sind. Diese strikte Berechnung kann jedoch bei der Analyse von Abläufen von zugriffssicheren Systemen nicht immer angewendet werden, denn durch eine Kombination der Zugriffe auf sicherheitsrelevante Daten kann ein erweiterter Schutz der Daten erforderlich werden. So muss beispielsweise in einem Versandunternehmen für einzelne Buchungseinträge nur die Integrität der Buchungsdaten gewährt werden. Die Vertraulichkeit der Daten muss bei reinen Buchungsdaten (wie etwa in einem Buchhandel ISBN-Nr., Preis des Buchs, Anzahl, etc.) nicht gewährt sein. Hingegen muss ein Zugriff auf alle Buchungsdaten, wie dieser im Rahmen einer Statistikauswertung notwendig ist, in Bezug auf die Vertraulichkeit geschützt werden, da andernfalls aus diesen Einzeldaten die Umsatzzahlen des Unternehmens offen gelegt werden können.

Da sich der Wert der Information durch die Verarbeitung ändert, kann mithilfe der transitiven Hülle über die Datenzugriffe nur das *Minimum* der Schutzziele für die Geschäftsprozesse aus den Geschäftsobjekten berechnet werden. Für eine vollständige Erfassung der Schutzziele müssen Schutzziele für *wertschöpfende Teilabläufe* gegebenenfalls manuell ergänzt werden.

Im Rahmen der Geschäftsprozessmodellierung annotieren wir eine Aktivität mit dem Schutzziel-Icon

- **Vertraulichkeit (Confidentiality)**, wenn bei der Ausführung der Aktivität kein unautorisierter Informationsgewinn möglich sein darf,
- **Integrität (Integrity)**, wenn bei der Ausführung der Aktivität Objekte nicht unbemerkt oder unautorisiert manipuliert werden dürfen, oder
- **Verbindlichkeit (Non Repudiation)**, wenn die Durchführung der Aktivität nicht abstreitbar sein darf.

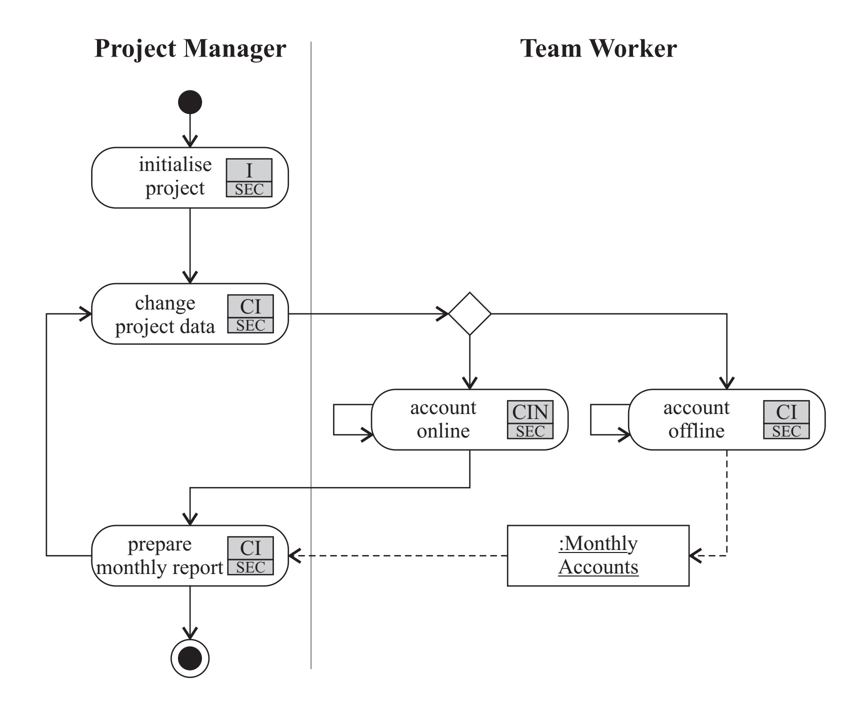


Abbildung 6.13: TimeTool-Geschäftsprozess mit Schutzzielen für Aktivitäten

Bereits in der Geschäftsprozessmodellierung werden so am Domänenmodell und an den Abläufen Schutzziele spezifiziert. Da jedoch die Modelle der Geschäftsprozessmodellierung noch nicht die endgültige Form erreicht haben, d.h., in einer für das Design geeigneten Form vorliegen, finden an diesen Modellen weitere Anpassungen und Verfeinerungen statt. Dies erfordert auch Anpassungen an den Schutzzielen und Benutzerrechten, die in den Iterationen durchzuführen sind. Die iterative Entwicklung wird unter anderem durch den iterativ anwendbaren Sicherheitsmikroprozess unterstützt (vgl. Abschnitt 5.3.3).

Abbildung 6.13 zeigt einen Geschäftsprozess unserer *TimeTool*-Fallstudie mit Schutzzielannotationen in den Aktivitäten. Zu den spezifizierten Schutzzielen sind im Rahmen der Entwicklung die Ausführungsrechte festzulegen sowie innerhalb des Sicherheitsmikroprozesses die Bedrohungen und Risiken zu ermitteln.

6.2.3 Schutzziele in Objektflüssen

Aktivitäten operieren über und auf Objekten. Sind Objekte, die in einem Aktivitätsablauf ausgetauscht werden, von besonderer Bedeutung, können diese auch in die Aktivitätsdiagramme mit aufgenommen werden (vgl. [OMG03, BRJ98]). Hierzu führen wir analog zu den Schutzziel-Stereotypen für Klassendiagramme Schutzziel-Stereotypen auf Flussobjekten ein.

Abbildung 6.14 zeigt diese Schutzziel-Stereotypen für Flussobjekte in Form von den bereits vorgestellten Schutzziel-Icons. Semantisch haben diese Schutzziel-Stereotypen dieselbe Bedeutung wie die Schutzziele zu den Geschäftsobjekten (vgl. Abschnitt 6.2.1). Innerhalb von UML-Aktivitätsdiagrammen werden jedoch keine Klassen verwendet, sondern konkrete Flussobjekte. Jedoch hat dies auf die Definition der Schutzziel-Stereotypen keinen Einfluss, sodass die Schutzziel-Stereotypen für Klassendiagramme direkt auf die Flussobjekte übernommen wer-

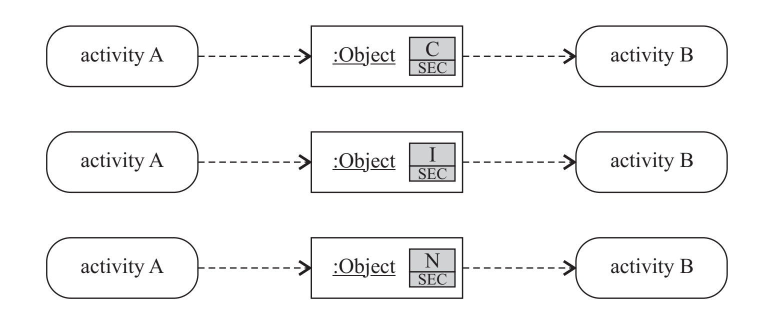


Abbildung 6.14: Schutzziele in Flussobjekten

den können. In Abbildung 6.14 sind die einzelnen Schutzziel-Icons dargestellt. Neben diesen einzelnen Schutzzielen ist eine Kombination der Schutzziele in einzelnen Objekten möglich.

Neben den vorgestellten Schutzzielen für Flussobjekte stellt das Schutzziel *Authentifikation* eine wichtige Rolle dar. Bei Systemen, die über ihre Systemgrenzen hinweg mit anderen Systemen kommunizieren, muss vor dem Empfang von Flussobjekten aus externen Systemen die Glaubwürdigkeit der Objekte anhand ihrer eindeutigen Identität oder charakteristischen Eigenschaften überprüft werden. Das kommunizierende System muss sich vorab authentifizieren, da andernfalls beliebige Software mit dem System interagieren könnte. Dieselbe Vorgehensweise gilt auch bei Systemen, die sich in mehrere Teilsysteme untergliedern und zwischen den Teilsystemen verschiedene Sicherheitsstufen existieren. Hier müssen Objekte, die aus einem weniger sicherheitskritischen Teil in einen kritischeren Teil Information übertragen, vorab bezüglich ihrer Authentizität überprüft werden.

Eng in Zusammenhang mit der Authentifikation steht in diesem Zusammenhang auch die Autorisierung, d.h., die Überprüfung der Benutzerrechte für den authentifizierten Nutzer. Im Rahmen der Betrachtung der Authentifikation in Objektflüssen unterscheiden wir jedoch nicht zwischen den beiden Begriffen. Eine detaillierte Trennung findet erst im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme statt (siehe hierzu Abschnitt 7.2).

In vielen Fällen reicht eine einseitige Authentifikation aus, d.h., der Sender eines Objektes authentifiziert sich beim Empfänger eines Objektes vor dem Senden des Objekts. Wir kennzeichnen dies durch ein Schutzziel-Icon mit dem Label *A* an der gestrichelten Linie des Objektflusses, wie dies in der oberen Hälfte von Abbildung 6.15 dargestellt ist.

Um jedoch sicherzustellen, dass der Sender mit dem gewünschten Server und nicht mit einem anderen Server kommuniziert, ist eine zweiseitige Authentifikation (engl. *mutual authentication*) [And01, Eck03, RE99] erforderlich. Wir kennzeichnen dies durch ein Schutzziel-Icon

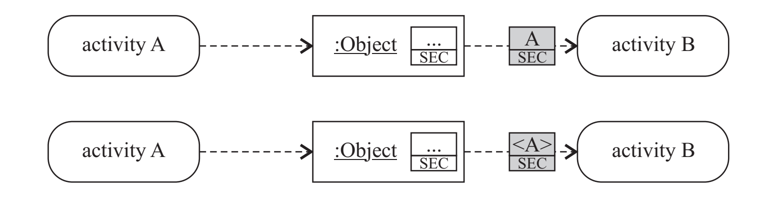


Abbildung 6.15: Schutzziel Authentifikation in Objektflüssen

mit dem Label $\langle A \rangle$ (vgl. Abbildung 6.15, untere Hälfte). Die zweiseitige Authentifikation ist beispielsweise in Zukunft im Bereich des Mobilfunks geplant. Derzeit antwortet ein mobiles Gerät immer derjenigen Basisstation, die das stärkste Signal aussendet. Errichtet ein Angreifer eine Basisstation mit einem stärkeren Signal, so authentifiziert sich eine mobile Station beim Angreifer.

Im Rahmen der Geschäftsprozessmodellierung wird die allgemeine Authentifikation neben der Authentifikation der Flussobjekte nicht weiter betrachtet. Eine detaillierte Betrachtung der Authentifikation findet im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme in Kapitel 7 statt.

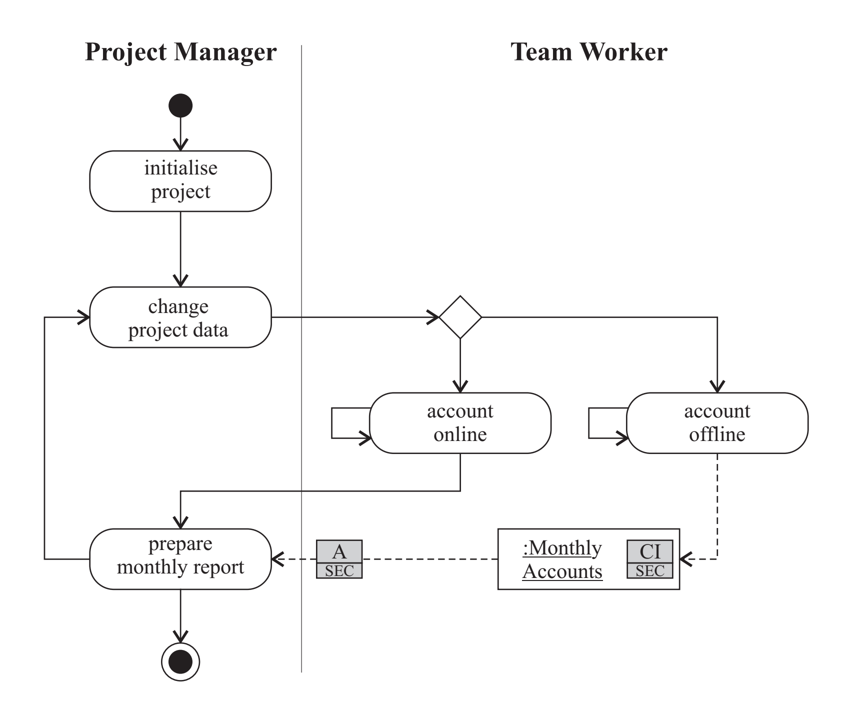


Abbildung 6.16: TimeTool-Geschäftsprozess mit Schutzzielen im Objektfluss

Abbildung 6.16 zeigt einen *TimeTool*-Geschäftsprozess mit Objektfluss. Stundenbuchungen können offline, beispielsweise auf einem Notebook, eingetragen werden. Vor der Übertragung dieser Buchungen auf den Server muss sich das auf dem Notebook ausgeführte Offline-Stundenbuchungssystem beim Server authentifizieren.

Abbildung 6.17 zeigt einen weiteren Geschäftsprozess aus unserer *TimeTool*-Fallstudie mit Betrachtung des Objektflusses. In diesem Geschäftsprozess wurden auch für manuelle Aktivitäten Schutzziele vergeben (zum Beispiel Aktivität *sum up testing time*). Dieses Beispiel zeigt, dass nicht nur automatisierbare Vorgänge mit den vorgestellten Techniken mit Schutzzielen annotiert werden können.

Bei der Entwicklung von zugriffssicheren Systemen ist zu beachten, dass die Integration von Sicherheit in Systemen auch angrenzende Arbeiten und Systeme betreffen muss. Was nützt die beste Sicherung von Daten im System, wenn diese außerhalb des Systems offen liegen und ungeschützt verarbeitet werden?

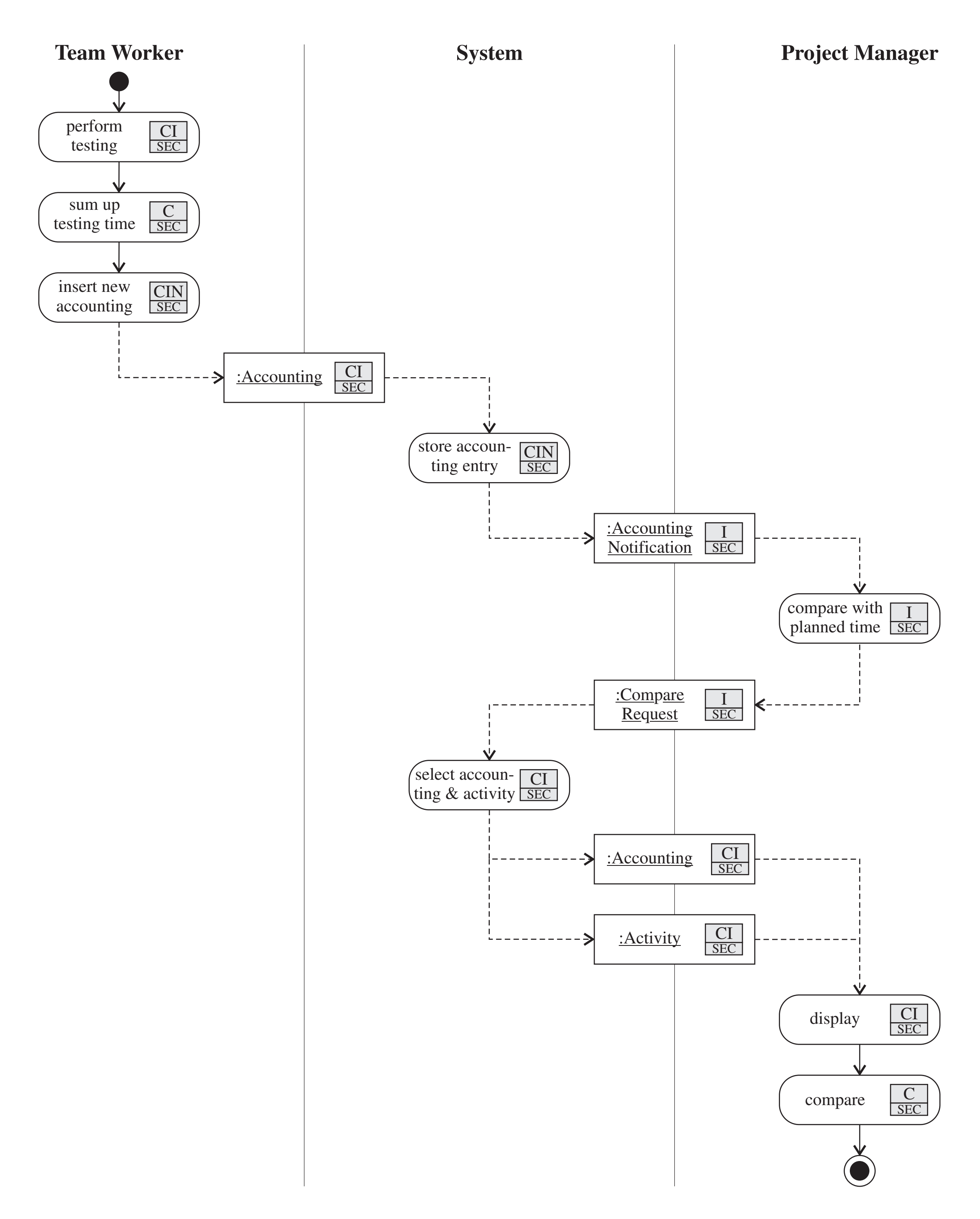


Abbildung 6.17: Objektfluss mit Sicherheitsanforderungen

6.3 Berechtigungsmodellierung

Die im Abschnitt 6.2 eingeführte Modellierung von Schutzzielen stellt eine Vorstufe zur Modellierung von Benutzerrechten dar. Dabei wurden Aktivitäten und Objekte gekennzeichnet, die potenzielle Angriffsziele darstellen. Neben der Untersuchung der tatsächlichen Bedrohun-

gen und Risiken dieser Aktivitäten und Objekte muss für diese Objekte und Aktivitäten festgelegt werden, wer auf diese zugreifen bzw. diese ausführen darf.

In diesem Abschnitt betrachten wir die Modellierung von Berechtigungen auf Objekten innerhalb der Geschäftsprozessmodellierung. Für die Modellierung von Berechtigungen wird in Abschnitt 6.3.1 vorab geklärt, welche Akteure in Form von Rollen es innerhalb des Systems gibt und welche Abhängigkeiten es zwischen den Akteuren gibt. Anschließend gehen wir in den Abschnitten 6.3.2 und 6.3.3 auf die Modellierung von Zugriffsrechten des Domänenmodells und Ausführungsrechten zu Aktivitäten ein. Die Berechtigungsmodellierung basiert auf dem in Kapitel 4 vorgestellten akteurzentrierten Modell zur Spezifikation von Benutzerrechten. Benutzerrechte unterteilen wir hier in zwei Kategorien – im Fall von Lese-, Schreib- und Create-Berechtigungen auf Domänenobjekten sprechen wir von Zugriffsrechten. Von Ausführungsrechten sprechen wir bei Execute-Berechtigungen auf Aktivitäten.

6.3.1 Modellierung der Akteure

Innerhalb der Geschäftsprozessmodellierung sind während der Entwicklung des Domänenmodells und der Geschäftsabläufe auch die Akteure des Systems, die auf den Daten arbeiten oder an den Abläufen beteiligt sind, zu identifizieren (vgl. [Rat00, Kru00]).

Für die Festlegung der Benutzerrechte ist im Rahmen der Geschäftsprozessmodellierung zugriffssicherer Systeme zu klären, welche Abhängigkeiten zwischen den Akteuren des Systems bestehen, sodass Benutzerrechte nicht redundant spezifiziert müssen und widerspruchsfrei definiert werden können. Werden für ähnliche Akteure die Benutzerrechte jeweils separat definiert, besteht die Gefahr, dass den Akteuren verschiedene Benutzerrechte zugeordnet werden. Um dies zu vermeiden, müssen Gemeinsamkeiten in den Rechten einmalig für alle ähnlichen Akteure festgelegt. Widersprüche können vermieden werden, wenn die Gemeinsamkeiten der Akteure ermittelt und für diese Gemeinsamkeiten die Rechte einmalig vergeben werden.

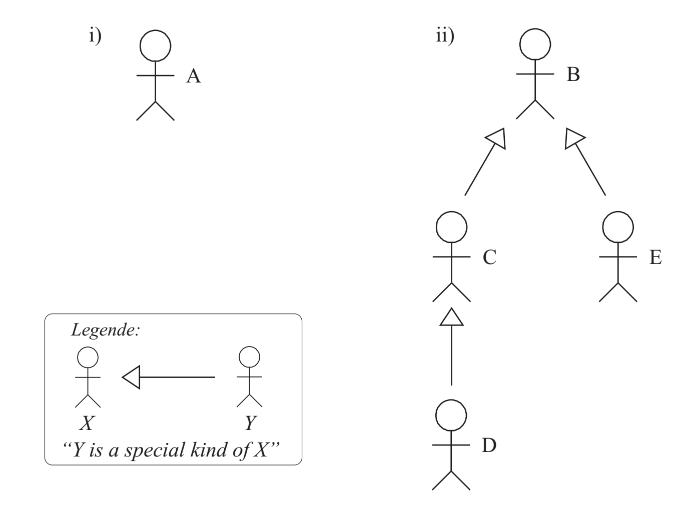


Abbildung 6.18: Hierarchie der Akteure

Die redundante Vergabe von Benutzerrechten für verschiedene Akteure mit gleichen Benutzerrechten ist ebenfalls zu vermeiden, da hier beispielsweise bei einer Änderung der Benutzerrechte ein einzelnes Benutzerrecht übersehen werden und dies zu Inkonsistenzen führen

kann. Um derartige Probleme bei der Modellierung der Benutzerrechte von vornherein auszuschließen, sind die Benutzer vorab in einer Hierarchie anzuordnen, sodass Gemeinsamkeiten ermittelt und die Benutzerrechte möglichst einheitlich vergeben werden können. Bei der Hierarchisierung sind zwei Abhängigkeitsvarianten möglich:

- Die Akteure existieren nebeneinander im System und haben eigene Benutzerrechte. Derartige Akteure sind nur von einem gemeinsamen, abstrakten übergeordneten Akteur abhängig, zwischen konkreten Akteuren gibt es keine Abhängigkeiten.
- Die Akteure sind hierarchisch angeordnet. Der speziellere Akteur *erbt* die Benutzerrechte von dem übergeordneten Akteur und besitzt selbst weitere eigene Benutzerrechte. Zudem können vererbte Benutzerrechte überschrieben werden, d.h., sie sind neu zu definieren (vgl. hierzu Vererbung von Benutzerrechten in Abschnitt 4.4.3).

Abbildung 6.18 zeigt ein Beispiel für eine mögliche Akteurhierarchie. Der Akteur *A* ist von den Akteuren *B, C, D, E* unabhängig. Die Akteure *C, E* sind eine spezielle Form des Akteurs *B*, der Akteur *D* ist zusätzlich eine Sonderform des Akteurs *C*, er erbt somit die Benutzerrechte der Akteure *B, C.*

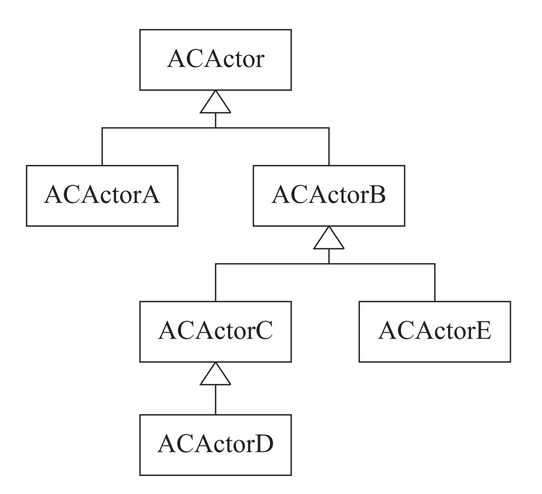


Abbildung 6.19: Hierarchiebildung in den Akteurklassen

In Abschnitt 4.4 wurde bereits die interne Repräsentation der Akteure mittels Akteur-Rollen vorgestellt. Dabei wurde für jede Akteur-Rolle des Systems eine Akteurklasse eingefügt. Zwischen den derart eingeführten Klassen lässt sich die Hierarchie in so genannten Vererbungsbäumen darstellen. Weiterhin wird eine abstrakte Superklasse *ACActor* eingeführt, die Basis aller Akteurklassen ist. In dieser Superklasse können allgemeine Berechtigungen, die für alle Akteure im System gelten sollen, eingefügt werden. Abbildung 6.19 zeigt die Akteurklassen zum Beispiel aus Abbildung 6.18.

Für unsere *TimeTool*-Fallstudie ist die Akteurhierarchie in Abbildung 6.20 dargestellt. Die Akteure *Administrator* und *Others* sind unabhängig zu den anderen Akteuren der Fallstudie. Der *ProjectManager* ist eine Sonderform des *TeamWorkers*, sodass für ihn seine eigenen Rechte sowie Rechte des *TeamWorkers* gültig sind, soweit diese nicht überschrieben, d.h., explizit definiert, wurden. Zusätzlich wurde hier noch der abstrakte, allen übergeordnete Akteur *Actor* dargestellt. Dieser abstrakte Akteur entspricht keiner Rolle. Dieser Akteur entspricht

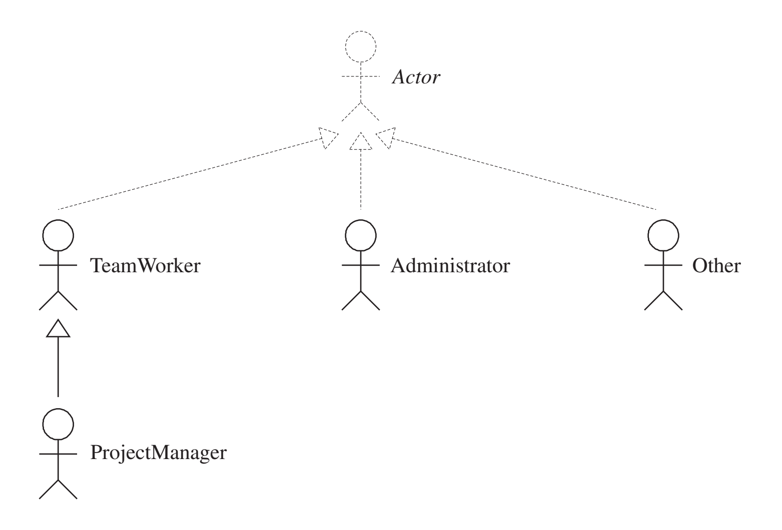


Abbildung 6.20: Hierarchie der Akteure der TimeTool-Fallstudie

der abstrakten Akteurklasse *ACActor*, in der allgemeine Berechtigungen, die für alle Akteure des Systems gelten, spezifiziert werden können.

In Literatur zur allgemeinen Geschäftsprozessmodellierung wird die Hierarchisierung von Personen und Rollen kaum behandelt. Zwar findet sich in ARIS (Architektur integrierter Informationssysteme) [Sch92, Sch99] innerhalb der Organisationsmodellierung eine hierarchische Anordnung der Organisation wieder, jedoch basiert diese nur auf Stellen oder Abteilungen, d.h., Personen oder Personentypen sind Abteilungen zugeordnet und diese werden wiederum Abteilungen zugeordnet. Die Abteilungen selbst können hierarchisch angeordnet werden. In [EP00] können Rollen durch Teamobjekte, die an Abteilungen assoziiert sind, dargestellt werden.

6.3.2 Modellierung von Zugriffsrechten auf das Domänenmodell

Die Zugriffsrechte auf Objekte des Geschäftsmodells beschreiben wir in der in Abschnitt 4.4 vorgestellten Weise. Diese leiten wir aus den Schutzzielannotationen in den Geschäfts- und Flussobjekten ab (vgl. hierzu Abschnitt 6.2), d.h., für alle mit Schutzzielen versehen Objekte sind Zugriffsberechtigungen zu erstellen. Die Zugriffsrechte werden entweder textuell für Klassen oder prädikativ für einzelne Zugriffsmethoden separat festgelegt. Durch die die Verwendung von Methodenkategorien können auch Zugriffsrechte für mehrere Zugriffsmethoden auf Attribute prädikativ beschrieben werden.

Zugriffsrechte können informal oder formal festgelegt werden. Da die Objekte innerhalb der Geschäftsprozessmodellierung oftmals in einer vorläufigen Form modelliert werden und innerhalb der weiteren Entwicklung, beispielsweise durch Ergänzungen oder Aufteilungen, noch verändert werden müssen, ist es sinnvoll, die Rechte in dieser frühen Phase vorab informal durch Text zu beschreiben. Ausnahmen können beispielsweise Objekte zur Datenspeicherung sein, deren Struktur nicht mehr verändert werden muss. Auch Schnittstellen zu externen Systemen können fest vorgegeben sein.

Ein weiterer Grund für eine vorab unpräzisere, textuelle Darstellung der Benutzerrechte liegt auch darin, dass zu diesem frühen Modellierungszeitpunkt die Attribute der Objekte teilweise noch nicht festgelegt sind.

Da sich die Modellierung von Geschäftsprozessen nicht nur auf die automatisierbaren Abläufe eines Unternehmens beschränkt, können auch Objekte spezifiziert werden, die sich in dem zu realisierenden System nicht wieder finden, d.h., die später nicht umgesetzt werden. Für derartige Objekte kann eine Spezifikation der Rechte jedoch ebenfalls sinnvoll sein, denn die Zugriffsrechte können auch Berechtigungen im manuellen Ablauf festlegen. Für manuelle Abläufe reicht jedoch die textuelle Spezifikation der Berechtigungen aus, da eine Generierung von Zugriffsmethoden hier nicht erforderlich ist. Eine präzise formale Spezifikation der Zugriffsrechte ist hier nicht möglich, wenn der Zusammenhang des manuell zu verarbeitenden Objekts mit der internen Objektstruktur der Geschäftsprozessmodellierung nicht festgelegt wurde.

Die Zugriffsrechte werden innerhalb einer Berechtigungsmatrix abgelegt, wie dies bereits im Kapitel 4 vorgestellt wurde.

6.3.3 Modellierung von Ausführungsrechten

Neben den Benutzerrechten auf Datenobjekten, die wir als Zugriffsrechte bezeichnen, sind bei der Rechtemodellierung auch die *Ausführungsrechte* festzulegen. Ausführungsrechte legen fest, welche Akteure die Berechtigung haben, einen Ablauf (in diesem Kapitel in Form von Ausführungsrechten für Aktivitäten, in späteren Kapiteln in Form von Ausführungsrechten für Anwendungsfälle) auszuführen. Dabei handelt es sich um die Festlegung von *Execute*-Berechtigungen auf Methoden, während Zugriffsrechte *Read*-, *Write*- oder *Create*-Berechtigungen auf Attribut-Zugriffsmethoden festlegen.

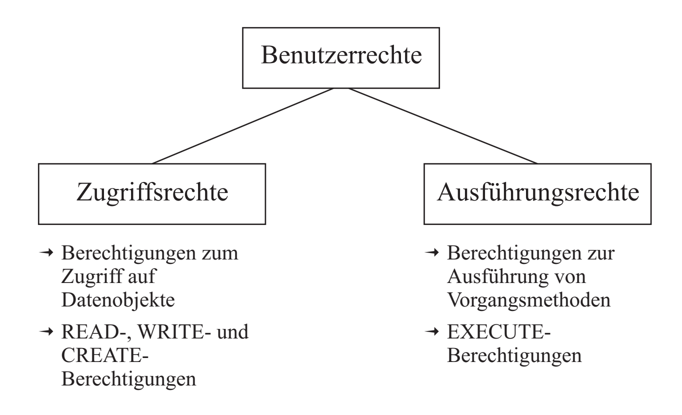


Abbildung 6.21: Unterscheidung von Benutzerrechten

Wie wir im Folgenden sehen werden, unterscheidet sich die Beschreibung von Ausführungsrechten in Syntax und Semantik nicht von der der Zugriffsrechte. Wir führen die begriffliche Trennung der Benutzerrechte in Zugriffs- und Ausführungsrechte nur zur Verdeutlichung ein, sodass begrifflich unterschieden werden kann, inwiefern es sich bei den Rechten um Zugriffsberechtigungen auf Daten oder auf Ausführungsberechtigungen zu Vorgangsmethoden zu Aktivitäten und Anwendungsfällen handelt. Abbildung 6.21 fasst die Unterscheidung zwischen Zugriffs- und Ausführungsrechten zusammen.

Bei der Entwicklung von zugriffssicheren Systemen wäre es in einigen speziellen Fällen möglich, die Zugriffssicherheit nur durch Zugriffsrechte auszudrücken. Die Beschreibung mittels

Zugriffsrechte reicht jedoch in folgenden Fällen nicht aus, sodass eine explizite Definition von Ausführungsrechten notwendig ist:

- Externer Code in Form von Systemfunktionen, DB-Funktionen, DB-Trigger, Applets, etc. ist auszuführen und der Zugriff hierzu ist zu beschränken.
- Durch die Kombination von Zugriffen innerhalb eines Ablaufs ändert sich der Wert der Information. So können beispielsweise einzelne Zugriffe auf Datenobjekte im System erlaubt sein. Der Zugriff auf alle Objekte einer Klasse lässt jedoch Schlussfolgerungen zu. Vorab unbedeutende Information, wie etwa eine Buchung in einem Warensystem, wird zu wertvoller Information, da sich Umsatzzahlen oder Ähnliches berechnen lassen. Derartige Zugriffe können nicht mehr in den einzelnen Zugriffsrechten ausgedrückt werden, sie müssen als Ausführungsrechte für die betreffenden Vorgangsmethoden spezifiziert werden.
- Sind Datenzugriffe und Aufrufe von Submethoden parametrisiert, so kann das zugeordnete Ausführungsrecht der aufrufenden Methode zu restriktiv oder unzureichend sein. Um einerseits alle potenziell unberechtigten Zugriffe abzuwehren, andererseits aber das Minimum an notwendigen Benutzerrechten zur Ausführung zu gewähren, müssen hier die Ausführungsrechte manuell festgelegt werden.
- Der Datenzugriff ist von Prinzipien abhängig. Beispielsweise darf aufgrund des *4-Augen-Prinzips* ein Vertrag, der zweimal signiert werden muss, nicht beide Male von derselben Person signiert werden. Derartige Ausführungsberechtigungen sind ebenfalls an Vorgänge zu binden.

Im Folgenden gehen wir näher auf die Modellierung von Ausführungsrechten ein und zeigen, wie wir den bereits für die Spezifikation von Zugriffsrechten gewählten Ansatz zur Beschreibung von Ausführungsrechten wieder verwenden. Bei der eigentlichen Modellierung stellen wir verschiedene Berechtigungstypen vor.

6.3.3.1 Zusammenhang zwischen Ausführungsrechten und Aktivitäten

Das im Kapitel 4 vorgestellte Rechtemodell spezifiziert Zugriffsberechtigungen auf Klassenmethoden. Wie wir in Abbildung 4.5 gezeigt haben, werden in unserem Berechtigungsansatz zur Benutzerrechtespezifikation Akteure auf *User*-Klassen abgebildet, sodass die Berechtigungen als Navigationen über die Objektstruktur ausgedrückt werden können.

Um einen homogenen Spezifikationsrahmen innerhalb P-MOS (vgl. Abschnitt 4.3) auch für Aktivitätsdiagramme bereitzustellen, erweitern wir das interne Klassendiagramm um Vorgangsklassen zur Darstellung der Aktivitäten.

Hierzu führen wir für jede Aktivität aus den Ablaufmodellierungen in Aktivitätsdiagrammen zum internen Klassendiagramm eine Vorgangsklasse *ProcessClass* hinzu. Jede dieser Vorgangsklassen enthält eine Methode *process*, welche die Ausführung der Aktivität zur Aufgabe hat. Wir setzen dabei a priori voraus, dass jede Aktivität im Aktivitätsdiagramm einer *Swimlane* und dass jeder *Swimlane* eindeutig eine Akteur-Rolle zugeordnet wurde.

Als Beispiel betrachten wir das Aktivitätsdiagramm in Abbildung 6.22. Die beiden Aktivitäten *Activity1* und *Activity2* sind jeweils einer *Swimlane* zugeordnet und den Swimlanes sind die

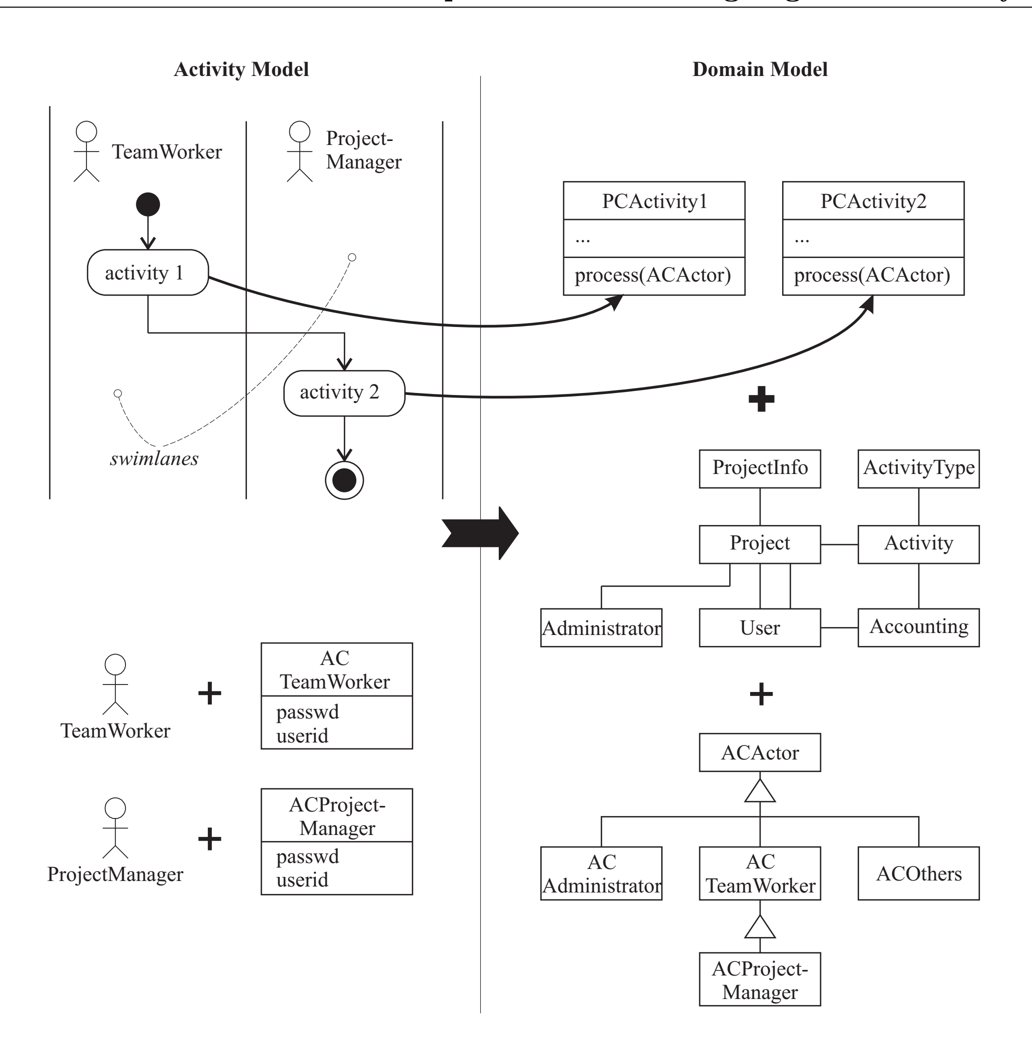


Abbildung 6.22: Erweiterung des Klassendiagramms um Vorgangsklassen

Akteur-Rollen *TeamWorker* und *ProjectManager* zugeordnet, welche die Aktivitäten *Activity1* bzw. *Activity2* ausführen. Zum internen Klassendiagramm der *TimeTool*-Fallstudie fügen wir die zwei Vorgangsklassen *PCActivity1* und *PCActivity2* hinzu, die jeweils eine Methode *process* enthalten.

Zu den derart abgeleiteten *process*-Methoden in den *Vorgangsklassen* können wir nun analog zu Zugriffsmethoden auf Klassenattribute *Zugriffsberechtigungen* in der in Abschnitt 4.4.2 vorgestellten Art

$$\textbf{\textit{funct}}\; perm_{\,C,m} : (ACActor, C, T_1, \ldots, T_n)\; Bool$$

vergeben, wobei C eine Klasse und m eine Methode von C der Form $m\text{-}id : (x_1 : T_1, \ldots, x_n : T_n)\, T$ ist. Die Eigenschaften von Methodenberechtigungen werden dabei durch P-MOS-Prädikate spezifiziert, die Bedingungen über der gegebenen Objektstruktur beschreiben.

Die Spezifikation von Ausführungsrechten ist nur deshalb möglich, da den Aktivitäten innerhalb den *Swimlanes* Akteur-Rollen zugeordnet sind. Diesen Akteur-Rollen wurden, wie in Abbildung 4.5 gezeigt, bereits *Akteurklassen* zugeordnet, sodass in den *process*-Methoden Akteur-spezifisiche Ausführungsrechte definiert werden können.

In Abbildung 6.22 sind auch nochmals die zum internen Klassendiagramm der *TimeTool*-Fallstudie hinzugefügten Akteurklassen *ACOthers*, *ACTeamWorker*, *ACProjectManager* und *ACAdministrator* dargestellt. In den *process*-Methoden der Vorgangsklassen *PCActivity1* und *PCActivity2* können Ausführungsberechtigungen für diese Akteure spezifiziert werden.

6.3.3.2 Modellierung der Ausführungsrechte

Die im vorausgehenden Abschnitt eingeführten Ausführungsrechte können für Rollen oder Individuen von Rollen vergeben werden. Weiterhin können Ausführungsrechte auch an Bedingungen über Individuen oder Rollen geknüpft werden. Im Folgenden diskutieren wir die verschiedenen Arten von Ausführungsrechten.

All-Role Permission

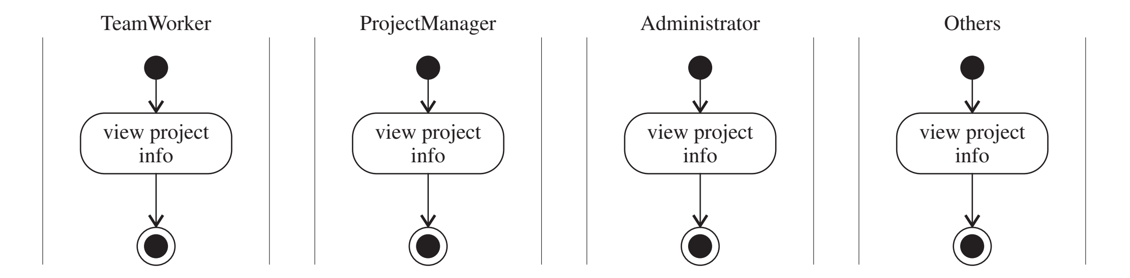


Abbildung 6.23: Beispiel zur unbeschränkten Ausführung

Aktivitäten im System, die keiner Einschränkung bezüglich der Ausführungsrechte unterliegen, sind die allgemeinste Form der Ausführungsrechte. Diese Aktivitäten dürfen unabhängig vom aktuellen Akteur ausgeführt werden. Da wir jedoch bei der Entwicklung von zugriffssicheren Systemen das Erlaubnisprinzip (engl. *fail-safe defaults principle*) zugrunde legen, ist grundsätzlich jeder Zugriff verboten und nur durch eine explizite Erlaubnis kann ein Zugriffsrecht gewährt werden. Aus diesem Grund müssen auch für Aktivitäten, die ohne Einschränkung ausgeführt werden können, Berechtigungen vergeben werden.

Abbildung 6.23 zeigt ein Beispiel für eine Aktivität ohne Beschränkung der Ausführung aus der *TimeTool*-Fallstudie. Die Aktivität *view project info* kann von jedem Akteur ausgeführt werden. Da eine Aktivität immer nur einer Rolle im Aktivitätsdiagramm zugeordnet werden kann, ist für jeden Akteur ein eigenes Aktivitätsdiagramm zu zeichnen. Die Abbildung zeigt vier Aktivitätsdiagramme für die verschiedenen Akteure der *TimeTool*-Fallstudie.

In der Benutzerrechtematrix ist ein unbeschränktes Ausführungsrecht dadurch gekennzeichnet, dass das Execute-Recht für alle Akteure als gegeben ist (Eintrag *true* in der Berechtigungsmatrix). Für die *TimeTool*-Fallstudie ergibt sich der folgende Ausschnitt der Benutzerrechtematrix:

actor → ↓ *class*	**Team Worker**	**Project Manager**	**Adminis-trator**	**Others**
PCViewProjectInfo	E: true	E: true	E: true	E: true

Aus den Ausführungsberechtigungen der Zugriffsmatrix lässt sich folgende Berechtigung für die Methode *process* der Vorgangsklasse *PCViewProjectInfo* formulieren:

$$\forall\, a : ACActor\ .\ \forall\, pc : PCViewProjectInfo$$
$$\Rightarrow\ perm_{\,PCViewProjectInfo,\,process}(a, pc)$$

Diese Berechtigung wurde für die Superklasse aller Akteure, für die Klasse *ACActor* definiert. Nach der in Abschnitt 4.4.3 eingeführten Vererbung von Benutzerberechtigungen gilt diese Berechtigung für alle vererbten und transitiv vererbten Akteur-Subklassen.

Swimlane-Permission

Eine Einschränkung der allgemeinen Ausführungsberechtigung ist die Beschränkung der Ausführung auf einzelne Akteure oder auf eine Teilmenge der Akteure. Die Aktivität darf nur noch von den explizit berechtigten Akteuren oder von Akteuren, die die notwendige Berechtigung erben, ausgeführt werden.

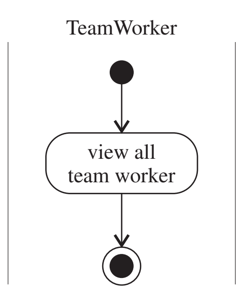


Abbildung 6.24: Beispiel zur Swimlane-Permission

Abbildung 6.24 zeigt die Aktivität *view all team worker* der *TimeTool*-Fallstudie. Diese Aktivität darf nur von Projektmitarbeitern ausgeführt werden. In der zugehörigen Zugriffsmatrix wird nur für die Rolle *TeamWorker* die Execute-Berechtigung auf *true* gesetzt.

actor → ↓ *class*	**Team Worker**	**Project Manager**	**Administrator**	**Others**
PCViewAllTeamWorker	E: true	E: –	E: –	E: –

Die derart definierten Berechtigungen gelten für alle Rollen, für die sie explizit vergeben werden, sowie für alle transitiven Rollen, die eine Sonderform dieser Rolle darstellen. Beispielsweise ist in der *TimeTool*-Fallstudie der *ProjectManager* eine Sonderform des *TeamWorkers* (vgl. Abbildung 6.20), der *ProjectManager* erhält somit implizit das Execute-Recht für die Methode *process* der Vorgangsklasse *PCViewProjectInfo*.

Zur Methode *process* der Vorgangsklasse *PCViewAllTeamWorker* ist folgende Permission zu definieren:

$$\forall\, tw : ACTeamWorker\ .\ \forall\, pc : PCViewAllTeamWorker$$
$$\Rightarrow\ perm_{\,PCViewProjectInfo,\,process}(tw, pc)$$

Im Gegensatz zur allgemeinen Permission aus dem vorigen Abschnitt wurde hier die Bedingung nur für den konkreten Akteur *ACTeamWorker* spezifiziert.

Objektstruktur-Permission

Eine weitere Beschränkung der Ausführung von Aktivitäten ergibt sich dadurch, dass die Ausführung an Bedingungen über Objekten oder Objektbeziehungen geknüpft ist. Eine Aktivität mit einer derart spezifizierten Berechtigung kann nur dann ausgeführt werden, wenn die vorab überprüften Attribut- und Referenzbelegungen von Objekten erfüllt sind.

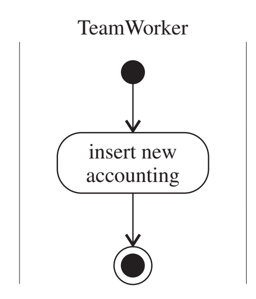


Abbildung 6.25: Beispiel zur Objektstruktur-Permission

Abbildung 6.25 zeigt ein Beispiel für eine Aktivität, die von der Objektstruktur abhängig ist. In unserer *TimeTool*-Fallstudie kann ein *TeamWorker* nur dann neue Buchungen einfügen, wenn die Buchung zu einem seiner Projekte gehört und wenn die Aktivität zum Buchen freigegeben wurde. In der folgenden Zugriffsmatrix ist diese Objektstrukturberechtigung textuell als Execute-Berechtigung formuliert:

↓ *class*	*actor* →	**TeamWorker**		**ProjectManager**	**...**
PCInsertNewAccounting	E:	insert new accountings to released activities from own projects	E:	–	...

Die Bedingung über der Objektstruktur tritt in der prädikativen Spezifikation der Execute-Permission auf. Bei der Spezifikation der Berechtigung der *process*-Methode der Vorgangsklasse *PCInsertNewAccounting* wird das hinzuzufügende Accounting-Objekt als Parameter übergeben.

$$\forall\, tw : ACTeamWorker \,.\, \forall\, a : Accounting \,.\, \forall\, pc : PCInsertNewAccounting \,.$$
$$a.user = tw.rolerep\,() \wedge a.activity.state = \sharp released$$
$$\Rightarrow\ perm_{\,PCInsertNewAccounting,\, process}(tw, pc, a)$$

Die Objektstrukturberechtigungen auf Aktivitäten stehen eng in Bezug mit Zugriffsberechtigungen auf Objekten. Finden innerhalb einer Aktivität nur Objektzugriffe und keine Aufrufe weiterer Ablaufmethoden aus Vorgangsklassen statt, so kann die Objektstruktur-Permission aus den Zugriffsberechtigungen der Objektzugriffsmethoden berechnet werden. Sie ist dann die logische *and*-Verknüpfung von allen Berechtigungen der Objektzugriffsmethoden, die in der *process*-Methode aufgerufen werden. Der Zusammenhang von Zugriffs- und Ausführungsberechtigungen wird in Abschnitt 7.4 diskutiert.

Instanz-Permission

Eine Sonderform der Objektstruktur-Permission stellt die Instanz-Permission dar. Hierbei darf eine Aktivität nur von bestimmten Instanzen der zugeordneten Rolle ausgeführt werden, d.h., nur von bestimmten Individuen als Rollenvertreter. Da die Rollenvertreter durch die Navigation auf den User-Klassen bestimmt werden können, handelt es sich bei der Instanz-Permission um eine Sonderform der Objektstruktur-Permission.

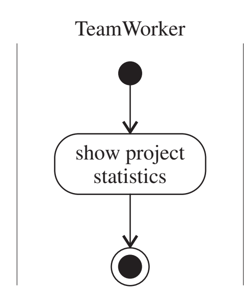


Abbildung 6.26: Beispiel zur Instanz-Permission

Abbildung 6.26 zeigt ein Beispiel aus der *TimeTool*-Fallstudie, wobei die Aktivität *show project statistics* nur von der konkreten Person *"Maier"* ausgeführt werden darf. Die Person *"Maier"* ist der Rolle *TeamWorker* zugeordnet. Hierzu ergibt sich folgender Eintrag in der Zugriffsmatrix:

↓ *class*	*actor* →	**TeamWorker**	**ProjectManager**	**...**
PCShowProjectStatistics	E:	only TeamWorker "Maier"	E: –	...

Da es sich bei der Instanz-Permission um einen Sonderfall der Objektstruktur-Permission handelt, wird diese analog zur Objektstruktur-Permission beschrieben:

$$\forall tw : ACTeamWorker \,.\, \forall pc : PCShowProjectStatistics \,.$$
$$tw.rolerep\,().name = \sharp Maier$$
$$\Rightarrow\; perm_{\,PCShowProjectStatistics,\, process}(tw, pc)$$

Auch in diesem Spezialfall kann die Ausführungsberechtigung der Aktivität aus den Ausführungsberechtigungen der *process*-Methode berechnet werden, wenn es sich bei den Zugriffsmethoden in der Aktivität ausschließlich um Objektzugriffe handelt.

Permission zur Instanzentrennung

Ein häufig wiederkehrendes Merkmal bei Geschäftsprozessen ist die Überprüfung der Arbeit (Kontrolle) durch eine weitere Person. So müssen Verträge zur Vermeidung von Fehlern oftmals von einer zweiten Person unterzeichnet werden, bevor das Dokument die Organisation oder das Unternehmen verlassen darf. Dieses so genannte *4-Augen-Prinzip* verlangt beispielsweise, dass in einem Vorgang zwei verschiedene Instanzen einer Rolle an einem Ablauf beteiligt sind.

Abbildung 6.27 zeigt ein Beispiel für einen Ablauf, in dem das 4-Augen-Prinzip angewendet werden muss. Bei dem dargestellten Szenario wird gefordert, dass ein Vertrag von zwei unterschiedlichen Projektmitarbeitern unterzeichnet wird.

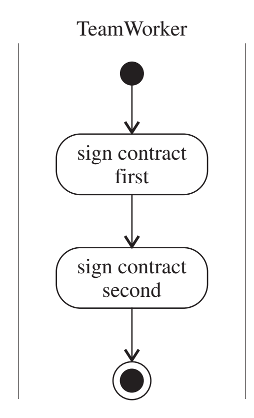


Abbildung 6.27: Beispiel einer Permissions zur Instanzentrennung

In der folgenden Zugriffsmatrix ist das *4-Augen-Prinzip* zum vorgestellten Szenario textuell spezifiziert dargestellt. Im Beispiel wird für die erste Aktivität angenommen, dass diese von jedem Projektmitarbeiter durchgeführt werden darf. Die zweite Aktivität darf dann von jedem anderen Vertreter der Rolle ausgeführt werden, mit Ausnahme der Instanz der Rolle der ersten Aktivität.

↓ *class* *actor* →	TeamWorker	ProjectManager	...
PCSignContractFirst	E: true	E: –	...
PCSignContractSecond	E: first and second signer have to be different team workers	E: –	...

Das *4-Augen-Prinzip* lässt sich durch das folgende Prädikat als Permission ausdrücken:

$$\begin{aligned}
&\forall\, tw : ACTeamWorker\ .\\
&\forall\, pc_1 : PCSignContractFirst\ .\ \forall\, pc_2 : PCSignContractSecond\ .\\
&\exists\, tw' : ACTeamWorker : pc_1.process\,(tw')\ \wedge\ tw'.rolerep\,() \neq tw.rolerep\,()\\
&\quad \Rightarrow\ perm_{\,PCSignContractSecond,\,process}(tw, pc_2)
\end{aligned}$$

Permission zur dynamischen Aufgabentrennung

Bei der dynamischen Aufgabentrennung dürfen innerhalb eines Ablaufs verschiedene Rollen nicht durch dieselbe Person vertreten werden. Der allgemein gültige Ansatz des Rollenparadigmas, dass eine Person in mehreren Rollen aktiv sein kann, wird hier eingeschränkt. Eine Person darf innerhalb eines Ablaufs nicht gleichzeitig in solchen Aktivitäten aktiv sein, deren assoziierten Aufgaben wechselseitig ausgeschlossen sind (vgl. [Eck03]).

Abbildung 6.28 zeigt ein Anwendungsszenario aus der *TimeTool*-Fallstudie zur dynamischen Aufgabentrennung. Eine Person kann sowohl Projektleiter als auch Projektmitarbeiter sein. Änderungen an den Stundenbuchungseinträgen eines Projektmitarbeiters müssen jedoch vom Projektleiter bestätigt werden. Damit ein Projektmitarbeiter nicht seine eigenen Änderungen zu Stundenbuchungen bestätigen kann, muss für diesen Ablauf die Rollenmitgliedschaft eingeschränkt werden: Der Projektleiter darf nicht seine eigenen Änderungen an Stundenbuchungen bestätigen.

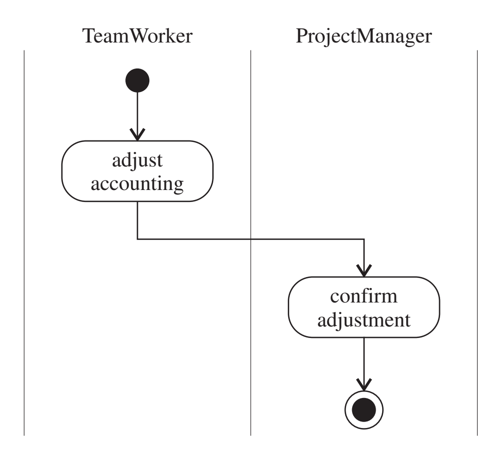


Abbildung 6.28: Szenario zur dynamischen Aufgabentrennung

↓ *class* *actor* →		**TeamWorker**		**ProjectManager**	**...**
PCAdjustAccounting	E:	all own accountings from released activities	E:	–	...
PCConfirmAdjustment	E:	–	E:	all accountings of activities of own projects where the project manager is not the team worker who has done the accounting adjustment	...

Die dynamische Aufgabentrennung lässt sich durch das folgende Prädikat ausdrücken:

$$\begin{aligned}
&\forall\, pm : ACProjectManager\ .\ \forall\, a : Accounting\ .\\
&\forall\, pc_1 : PCAdjustAccounting\ .\ \forall\, pc_2 : PCConfirmAdjustment\ .\\
&a.activity.project.projectmanager = pm.rolerep\,()\ \wedge\\
&\exists\, tw : ACTeamWorker : pc_1.process\,(tw)\ \wedge\ tw.rolerep\,() \neq pm.rolerep\,()\\
&\quad \Rightarrow\ perm_{\,PCConfirmAdjustment,\, process}(pm, pc_2, a)
\end{aligned}$$

Die Berechtigung des Projektmitarbeiters zur Aktivität *adjust accounting* wurde bereits bei der Objektstruktur-Permission eingeführt. Sie ist hier Voraussetzung für die dynamische Aufgabentrennung.

Anmerkung zur statischen Aufgabentrennung

Neben der dynamischen gibt es auch die statische Aufgabentrennung. Hierbei wird die Rollenmitgliedschaft nicht nur für einen einzelnen Ablauf beschränkt, sondern die hier festgelegte Aufgabentrennung gilt global. Bei der statischen Aufgabentrennung wird die gleichzeitige Mitgliedschaft in verschieden Rollen ausgeschlossen. Eine Person darf nur dann Mitglied unterschiedlicher Rollen sein, wenn sich die den Rollen assoziierten Aufgaben wechselseitig nicht ausschließen (vgl. [Eck03]).

Abbildung 6.29 zeigt ein Szenario zur statischen Aufgabentrennung. Wie aus dem Ablaufdiagramm hervorgeht, darf ein Projektmitarbeiter kein Administrator des Systems sein, denn dadurch würde er sich im *TimeTool*-System die Möglichkeit verschaffen, Stundenbuchungen

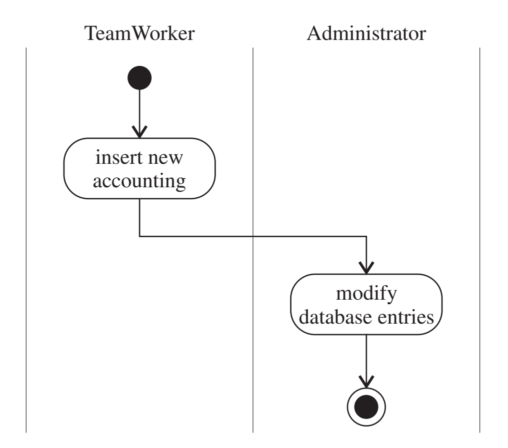


Abbildung 6.29: Szenario zur statischen Aufgabentrennung

und andere zu schützende Elemente direkt innerhalb der Datenbank zu manipulieren. Von weiterführenden Techniken, die auch einen Administrator die Manipulation verhindern, abstrahieren wir in Zusammenhang mit diesem Beispiel.

In unserem Modell gehen wir davon aus, dass Berechtigungen zur statischen Aufgabentrennung bereits während der Authentifikation überprüft werden. Verstoßen einzelne Rollenvertreter gegen diese globalen Berechtigungen, so schlägt die Anmeldung fehl. Auf die Authentifikation gehen wir im Rahmen der Anwendungsfallmodellierung in Kapitel 7 näher ein.

6.4 Der Prozess der Geschäftsprozessmodellierung zugriffssicherer Systeme

Nachdem wir uns in den vorausgehenden Abschnitten mit Beschreibungstechniken zur Annotation von Schutzzielen und der Darstellung von Benutzerrechten beschäftigt haben, stellen wir im Abschnitt 6.4.1 das Prozessmuster zum Ablauf der Geschäftsprozessmodellierung zugriffssicherer Systeme vor. Abschnitt 6.4.2 zeigt die exemplarische Anwendung des Sicherheitsmikroprozesses für die Modellierung von Bedrohungen und Risiken.

6.4.1 Prozessmuster 6.1: Geschäftsprozessmodellierung zugriffssicherer Systeme

Kurzbeschreibung

Zu Beginn einer Systementwicklung sind die Daten und Abläufe der Organisation bzw. Unternehmung zu erarbeiten. Für eine durchgängige Entwicklung der Sicherheit müssen bereits zu diesem Zeitpunkt Aspekte der Zugriffssicherheit betrachtet werden.

In diesem Prozessmuster zeigen wir die frühzeitige Modellierung von Aspekten der Zugriffssicherheit in Struktur und Verhalten. Neben der eigentlichen Modellierung von Aspekten der Zugriffssicherheit zeigen wir die Eingliederung dieser Tätigkeiten in die allgemeine Geschäftsprozessmodellierung. Die Aktivitäten der allgemeinen Geschäftsprozessmodellierung umreißen wir knapp, auf die Aktivitäten zur Modellierung zugriffssicherer Systeme gehen wir näher ein.

Problem

Bei der Geschäftsprozessmodellierung werden die Struktur und das Verhalten der Organisation bzw. Unternehmung analysiert und beschrieben, in welche das zu entwickelnde System eingebettet wird. Probleme der Organisation bzw. Unternehmung sind herauszuarbeiten und gegebenenfalls sind Verbesserungsmöglichkeiten aufzuzeigen. Es ist ein gemeinsames Verständnis zwischen Anforderungsentwicklern, Anwendern, Auftragnehmern und Auftraggebern zu erarbeiten und die Systemanforderungen der Zielplattform sind zu klären.

Innerhalb der allgemeinen Geschäftsprozessmodellierung werden in dieser frühen Phase keine Sicherheitsaspekte betrachtet, obwohl bei der Analyse von Domänenobjekten und Geschäftsprozessen Aspekte der Zugriffssicherheit schon betrachtet werden können. Bei der Modellierung des Domänenmodells können beispielsweise die bereits ermittelten Daten bezüglich potenzieller Angriffe in Bezug auf Vertraulichkeit, Integrität und Verbindlichkeit der Daten untersucht werden.

Bei der Evaluierung der gegenwärtigen Prozesse in Hinblick auf eine Automatisierung bleiben ebenfalls Aspekte der Zugriffssicherheit außen vor. Im Vordergrund für Änderungen der Prozesse stehen allgemeine Verbesserungen und Anpassungen, eine Optimierung aufgrund von Sicherheitsbetrachtungen findet nicht statt. Dies kann dazu führen, dass ein zu ändernder Prozess bei einer späteren Integration von Sicherheitsaspekten erneut geändert und optimiert werden muss.

Weiterhin wird in den heute üblichen Verfahren zur Geschäftsprozessmodellierung (siehe zum Beispiel [Bal98, EP00, Kru00, DW98]) das gemeinsame Wissen über Sicherheitsanforderungen nicht ausgenützt. Obwohl zu Beginn Anforderungsentwickler, Auftragnehmer, Auftraggeber und Anwender gemeinsam die Rahmenbedingungen und Anforderungen festlegen, werden existierende Lösungen in der manuellen oder automatisierten Abarbeitung zur Gewährleistung der Sicherheit nicht ausgenützt. Beispielsweise werden hier vertrauenswürdige Daten oder Aktivitäten, deren Ausführung im Nachhinein nicht abstreitbar sein dürfen, nicht gekennzeichnet. In späteren Phasen der Systementwicklung beginnen die Anforderungsentwickler, Aspekte der Zugriffssicherheit von Grund auf zu erarbeiten.

Die Benutzerrechte auf grundlegende Domänenobjekte und Geschäftsprozesse werden derzeit innerhalb der frühen Phasen der Softwareentwicklung nicht modelliert. Dies liegt unter anderem daran, dass die derzeitigen Ansätze zur Modellierung von Benutzerrechten Rechte zumeist in Form von Klassen im Klassendiagramm modellieren [BKL01, LBD02, BDL03, KPP03, HJ03, AW03b, Alt04]. Da jedoch das Klassendiagramm zu diesem Zeitpunkt noch *instabil* ist, d.h., die Struktur des Klassendiagramms oder einzelne Attribute können sich noch ändern, bleibt eine Modellierung der Benutzerrechte zu diesem Zeitpunkt außen vor.

Lösung

Für die frühzeitige Behandlung von Aspekten der Zugriffssicherheit innerhalb des Entwicklungsprozesses finden neben den Aktivitäten der allgemeinen Geschäftsprozessmodellierung aus [Kru00] Aktivitäten zur Modellierung von Schutzzielen sowohl in der Struktur der Domänenobjekte sowie in den Geschäftsprozessen statt. Akteure werden bezüglich der Rechte im System hierarchisch angeordnet, sodass Benutzern bezüglich ihres Rangs im Unternehmen verschiedene Rechte zugeteilt werden können. Basierend auf den ermittelten Schutzzielen in den Aktivitäten der Geschäftsprozesse und den Objekten des Domänenmodells sind die

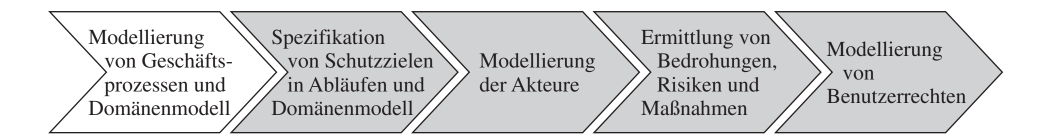


Abbildung 6.30: Die Aktivitäten der Geschäftsprozessmodellierung zugriffssicherer Systeme

potenziellen Bedrohungen zu ermitteln und die Risiken als Eintrittswahrscheinlichkeit der Bedrohung sind zu bestimmen. Weiterhin sind geeignete Maßnahmen für eine Gegenabwehr zu finden und diese Maßnahmen sind auf ihre Tauglichkeit zu überprüfen. Abbildung 6.30 gibt einen Überblick über die Aktivitäten der Geschäftsprozessmodellierung zugriffssicherer Systeme. Die Aktivitäten, die sich mit der Analyse der Sicherheitsaspekte befassen, sind dabei grau hinterlegt.

Besondere Bedeutung bei der Ausführung der Aktivitäten zur Geschäftsprozessmodellierung zugriffssicherer Systeme liegt dabei auf der Zusammenarbeit von Anforderungsentwicklern, Auftragnehmern, Auftraggebern und Anwendern bei der Bestimmung der Sicherheitsaspekte. Sowohl bei der Ermittlung der sicherheitskritischen Aktivitäten als auch bei der Identifizierung der schutzbedürftigen Objekte werden herkömmliche Bearbeitungsschritte und Dokumente analysiert. Der Umgang der Anwender mit den Dokumenten sowie der genaue betriebsinterne Ablauf sind wichtige Informationen für die Behandlung der Abläufe und Daten im System. Weiterhin können zusätzliche Sicherheitsanforderungen, die durch die Einführung eines Systems entstehen, von den Anforderungsentwicklern mit eingebracht und in die Modellierung aufgenommen werden.

Bei der Ermittlung der Bedrohungen können ebenfalls durch den gewohnten Ablauf und Umgang mit Daten potenzielle Angriffsmöglichkeiten ermittelt werden. Entscheidend ist auch die Mitwirkung der Anwender und Auftraggeber bei der Bestimmung der Benutzerrechte.

Die Benutzerrechte der Akteure zum Zugriff auf die Domänenobjekte sowie zur Ausführung von Aktivitäten der Geschäftsprozesse werden außerhalb des Klassendiagramms zunächst textuell und später formal innerhalb einer Benutzerrechtematrix modelliert. Dies hat zum einen den Vorteil, dass Benutzerrechte nicht bei jeder kleinen Änderung des Klassendiagramms geändert werden müssen, wenn sich beispielsweise Assoziationen ändern oder Klassen neu strukturiert werden. Weiterhin können die Benutzerrechte vorab bei der gemeinsamen Erarbeitung gemeinsam mit den Endanwendern, Anforderungsentwicklern, Auftragnehmern und Auftraggebern rein textuell spezifiziert werden. Für eine automatische Codegenierung der Benutzerrechte sind diese im Verlauf der weiteren Entwicklung zu formalisieren. Objekte und Aktivitäten, die innerhalb der Geschäftsprozessmodellierung bereits *stabil* sind, d.h., für die keine Änderungen zu erwarten sind, können bereits innerhalb dieser Phase formalisiert werden, während *instabile* Klassen und Aktivitäten erst in den weiteren Teilabschnitten der Anforderungsspezifikation zugriffssicherer Systeme *verfeinert* werden.

Aktivitäten

Bei der Prozessausführung der Geschäftsprozessmodellierung zugriffssicherer Systeme sind die folgenden Aktivitäten nach der in der Lösung und in der Struktur skizzierten Vorgehensweise auszuführen. Zur Vollständigkeit wurden bei den Aktivitäten auch die Aktivitäten der allgemeinen Geschäftsprozessmodellierung aufgenommen und kurz umrissen.

- *Identifikation der Geschäftsprozesse*
  Gemeinsam im Team aus Auftraggeber, Auftragnehmer, Anforderungsentwickler und Anwender werden die Grenzen der zu beschreibenden Organisation, Unternehmung oder Abteilung erarbeitet. Nach der Festlegung der Terminologie werden die Geschäftsprozesse skizziert und priorisiert.

- *Verfeinerung der Geschäftsprozesse*
  Die skizzierten Geschäftsprozesse werden ausgearbeitet und einem Review unterzogen. In diesem Review sind erneut Vertreter der Anforderungsentwickler, Auftraggeber, Auftragnehmer und Anwender beteiligt. Im Review wird geprüft, inwiefern die spezifizierten Geschäftsprozesse den Abläufen der Organisation bzw. Unternehmung entsprechen.

- *Geschäftsprozesslösungen entwerfen*
  Für eine genaue Festlegung der Geschäftsprozesse sind die beteiligten Rollen, die zu erstellenden Produkte (Software, Hardware), die Auslieferungen und alle potenziellen Vorfälle der zu modellierenden Unternehmung bzw. Organisation auszuarbeiten. Es wird festgelegt, welche Akteure die Geschäftsprozesse ausführen.

- *Verfeinerung der Rollen und Verantwortlichkeiten*
  Die Hardware-/Software-Produkte und die Zuständigkeiten der Bearbeiter sind genau zu definieren. Weiterhin ist zu überprüfen, inwiefern die Ergebnisse (hier die Geschäftsobjekte und Abläufe) den Vorstellungen der Auftraggeber entsprechen.

- *Domänenmodellierung*
  Ein *unvollständiges* Domänenmodell (vgl. [Kru00]) mit strategisch wichtigen Informationen (siehe [EP00]) ist auszuarbeiten. Dabei sind die grundlegenden Daten, auf denen die Geschäftsprozesse operieren, zu erfassen. Von internen Details, wie beispielsweise die Repräsentation der Daten innerhalb einer Datenbank, wird abstrahiert. Es wird ein Klassendiagramm mit den wichtigsten Informationsquellen erstellt.

- *Untersuchung der Prozessautomatisierung*
  Die bisher identifizierten und verfeinerten Geschäftsprozesse beschreiben die Geschäftsprozesse einer Unternehmung. Nicht alle Prozesse sind für eine Automatisierung geeignet, denn bestimmte Aufgaben können nicht oder nur schwer von einem System übernommen werden. Die Geschäftsprozesse sind deshalb dahingehend zu untersuchen, inwiefern sie sich für eine Automatisierung eignen. Die Prozessautomatisierung sind existierende Lösungen heranzuziehen, die Teilaufgaben lösen und in den Lösungsprozess integriert werden können. Aus den Ergebnissen der Untersuchung zur Prozessautomatisierung sind Systemanforderungen abzuleiten.

- *Spezifikation der Schutzziele in den Geschäftsprozessen*
  An den spezifizierten Geschäftsprozessen sind die Aktivitäten, die ein potenzielles Angriffsziel darstellen, mit Schutzzielen zu annotieren. Im Abschnitt 6.2.2 wurden hierzu spezielle Schutzziel-Icons für Aktivitätsdiagramme eingeführt, sodass potenzielle Angriffe an einzelnen Aktivitäten dieser Aktivitätsdiagramme annotiert werden können.

  Bei der Spezifikation der Schutzziele in den Geschäftsprozessen wird jede Aktivität aus jedem Aktivitätsdiagramm dahingehend untersucht, inwiefern bei der Abarbeitung dieser Aktivität ein unautorisierter Informationsgewinn, eine unbemerkte oder unautorisierte Manipulation zu vermeiden ist oder eine Beweissicherung notwendig ist.

Da der genaue interne Ablauf der Aktivitäten zu diesem Zeitpunkt noch nicht spezifiziert wurde, müssen bei dieser Spezifikation der Schutzziele die Daten oder Systemfunktionen, auf denen die aktuelle Aktivität operiert, in einer ersten Version ermittelt werden. Weiterhin sind die potenziellen Angriffsziele auch mit den Auftraggebern und Anwendern zu diskutieren. Schutzziele sind, falls möglich, aus der manuellen Ausführung des Geschäftsablaufs zu bestimmen. Oftmals werden vertrauliche Daten in Schränken eingesperrt oder nichtabstreitbare Dokumente von beiden Parteien durch ihre Unterschriften bestätigt.

- *Spezifikation der Schutzziele in den Objekten des Domänenmodells*
  Parallel zur Spezifikation der Schutzziele in den Geschäftsprozessen sind potenzielle Angriffsziele in den Objekten des Domänenmodells als Schutzziele zu annotieren. Falls in den Domänenobjekten vertrauliche, verbindliche oder integrielle Daten gelesen, geschrieben oder erzeugt werden, sind diese mit den in Abschnitt 6.2.1 eingeführten Schutzziel-Stereotypen zu annotieren.

- *Ausführung des Sicherheitsmikroprozesses*
  Bei der Ausführung des Sicherheitsmikroprozesses sind für jedes annotierte Schutzziel sowohl in den Geschäftsprozessen als in den Domänenobjekten Bedrohungen, Risiken und Maßnahmen zu suchen. Einzelheiten zum Sicherheitsmikroprozess sind im Prozessmuster zum Sicherheitsmikroprozess in den Abschnitten 5.3.3 und 6.4.2 beschieben.

- *Modellierung der Akteurhierarchie*
  Für die Festlegung der Benutzerrechte zum Ausführen von Aktivitäten (Ausführungsrechte) und zum Zugriff auf Daten (Zugriffsrechte) müssen die Akteure angeordnet werden. Die innerhalb der oben beschriebenen Aktivitäten “Geschäftsprozesslösungen entwerfen” und “Verfeinerung der Rollen und Verantwortlichkeiten” ermittelten Akteure sind, wie im Abschnitt 6.3.1 gezeigt, hierarchisch anzuordnen, sodass die Benutzerrechte widerspruchsfrei und ohne Redundanzen spezifiziert werden können.

- *Modellierung der Ausführungsrechte*
  Für die ermittelten und hierarchisch angeordneten Akteure sind die Ausführungsrechte auf Aktivitäten zu spezifizieren. Alle Aktivitäten, die vorab mit einem Schutzziel annotiert wurden, sind dabei einer genauen Analyse zu unterziehen. Den verschiedenen Akteuren des Systems sind aufbauend auf den Analyseergebnissen Ausführungsrechte zu vergeben. Für alle Aktivitäten, die mit keinem Schutzziel versehen wurden, ist die Ausführung nicht beschränkt. Da der Benutzerrechtespezifikation das Erlaubnisprinzip zugrunde liegt, müssen jedoch den Akteuren zur Ausführung von nicht beschränkten Aktivitäten die notwendigen Benutzerrechte eingeräumt werden.

  Da innerhalb der Geschäftsprozessmodellierung viele Einzelheiten noch offen sind, können sich die Ausführungsrechte während der weiteren Analyse noch verändern. Aus diesem Grund werden die Ausführungsrechte vorab textuell beschrieben. Sobald die Ausführungsrechte stabil sind, d.h., keine Änderungen mehr zu erwarten sind, werden sie in Hinblick auf eine automatische Generierung der Zugriffsmethoden prädikativ formalisiert. Für diejenigen Aktivitäten des Systems, für die bereits innerhalb der Geschäftsprozessmodellierung die Ausführungsrechte stabil sind, können auch bereits hier diese Rechte formal spezifiziert werden. Jedoch ist zu erwarten, dass im Rahmen der Geschäftsprozessmodellierung nur ein kleiner Teil der Ausführungsrechte formal

spezifiziert werden kann. Die Modellierung von Ausführungsrechten ist im Detail im Abschnitt 6.3.3 beschrieben.

- *Modellierung der Zugriffsrechte*
  Neben den Ausführungsrechten sind auch die Zugriffsrechte auf die Domänenobjekte zu spezifizieren. Alle mit Schutzzielen annotierten Klassen sind einer Benutzerrechtemodellierung zu unterziehen. Auch hier sind die Rechte vorab textuell und in Hinblick auf eine automatische Generierung formal zu spezifizieren. Für Objekte ohne Schutzziele sind die Zugriffsrechte nicht beschränkt. Da unsere Benutzerrechtespezifikationen auf dem Erlaubnisprinzip (vgl. Abschnitt 2.2.3) basiert, muss für derartige Objekte und Aktivitäten der allgemeine Zugriff explizit erlaubt werden. Die Modellierung von Zugriffsrechten wurde bereits in Abschnitt 6.3.2 beschrieben.

Produktartefakte

Die folgende Auflistung beschreibt die Eingaben und Ausgaben zum Prozess der Geschäftsprozessmodellierung zugriffssicherer Systeme. Um Sicherheitsaspekte erweiterte Produktartefakte der allgemeinen Geschäftsprozessmodellierung sind gekennzeichnet.

Eingabe-Produktartefakte

- Projektidee
- Projektmission und Sicherheitsziel (aus der Startphase)

Ausgabe-Produktartefakte

- Domänenmodell (mit Sicherheitsaspekten erweitert)
- Geschäftsprozessmodell (mit Sicherheitsaspekten erweitert)
- Bedrohungs- und Risikomodell
- Benutzerrechtemodell

Kontext

Dieses Prozessmuster kann nur im Rahmen einer Anforderungsspezifikation zugriffssicherer Systeme (vgl. Abschnitt 5.3.2), basierend auf einem objektorientierten Vorgehensmodell, ausgeführt werden. Der Fokus dieser Geschäftsprozessmodellierung zugriffssicherer Systeme liegt auf einer Neuentwicklung eines Systems. Aktivitäten zur Analyse bestehender Systeme sind kein Bestandteil dieses Prozessmusters.

Neben den Entwicklungsschritten sind auch projektübergreifend Managementaufgaben durchzuführen, auf die jedoch im Rahmen dieser Arbeit nicht näher eingegangen wird.

Struktur

Innerhalb der Lösung wurde bereits der Ablauf der Geschäftsprozessmodellierung zugriffssicherer Systeme skizziert. Im Aktivitätsdiagramm in Abbildung 6.31 ist der verfeinerte Ablauf dargestellt. Zudem sind auch mögliche Iterationsmöglichkeiten mit aufgenommen worden.

Einzelne Geschäftsprozesse oder Teile des Domänenmodells können separat entwickelt und dabei sofort mit Schutzzielen annotiert werden. Für diese inkrementelle Erarbeitung der Geschäftsprozesse und -objekte wurde im Ablauf zur Geschäftsprozessmodellierung zugriffssicherer Systeme eine Iterationsmöglichkeit eingefügt.

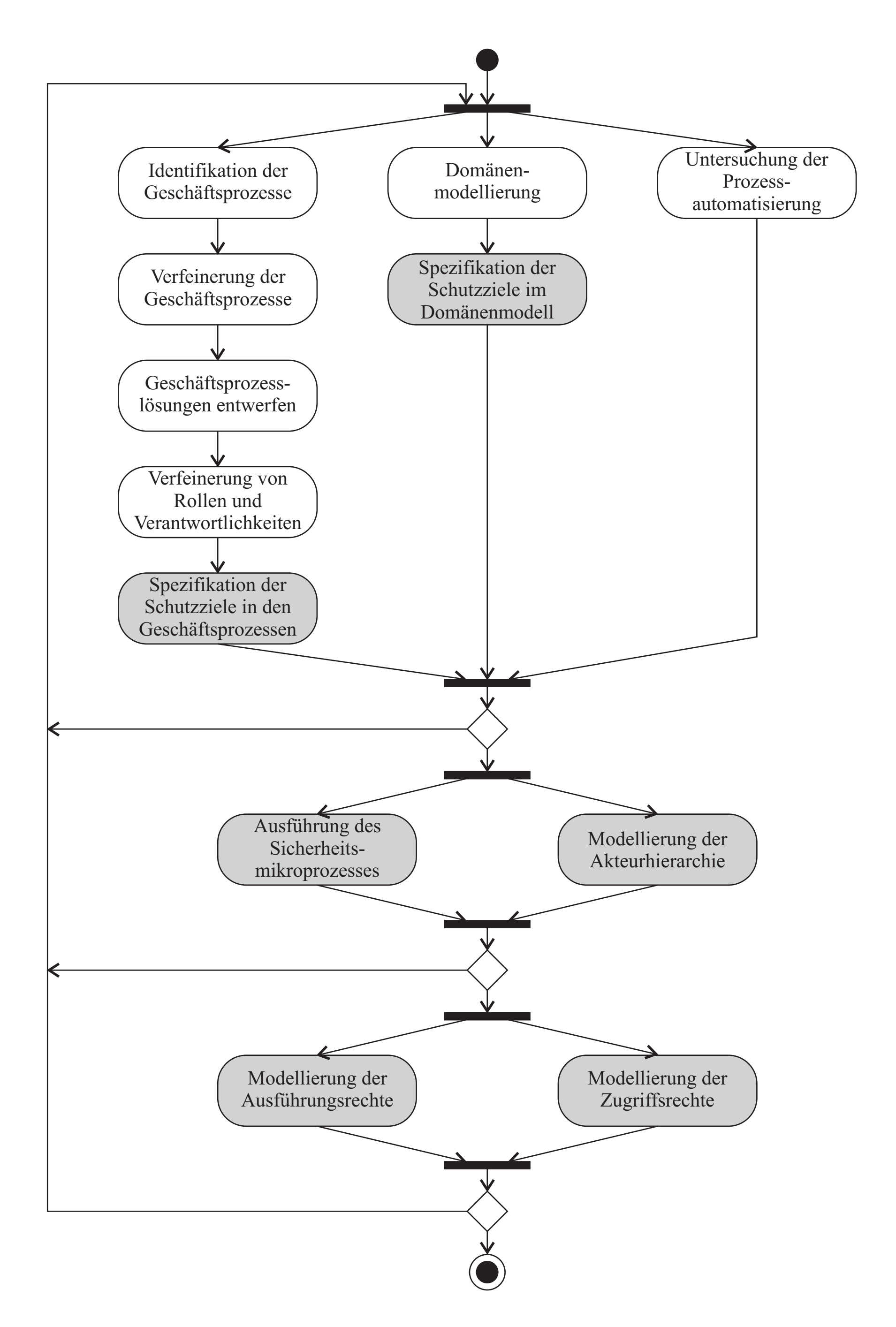


Abbildung 6.31: Der Ablauf der Geschäftsprozessmodellierung zugriffssicherer Systeme

Die nächste Stufe, die Ermittlung von Bedrohungen und Risiken sowie die Hierarchisierung der Akteure kann entweder nach Ermittlung aller Geschäftsprozesse und des gesamten Domänenmodells stattfinden oder aber nur an einem Teil mit anschließender inkrementeller Vervollständigung. Dieselbe iterative Entwicklung kann auch für die Modellierung der Benutzerrechte angewendet werden. Diese können ebenfalls einmalig oder mehrfach in Inkrementen spezifiziert werden.

Vor- und Nachteile

Mit dem vorgestellten Prozess wird die Entwicklung von Aspekten der Zugriffssicherheit frühestmöglich begonnen. Bereits bei der Definition der Prozesse, wie sie im Unternehmen bzw. in der Organisation ablaufen, werden potenzielle Angriffsmöglichkeiten festgehalten, hierzu die Bedrohungen und Risiken ermittelt, Maßnahmen ausgewählt und die Benutzerrechte spezifiziert.

Ein Vorteil dieses Verfahrens besteht darin, dass Auftraggeber, Auftragnehmer, Anforderungsentwickler und Anwender gemeinsam an der Entwicklung von Aspekten der Zugriffssicherheit beteiligt sind. So können auch die gewöhnlichen manuellen Vorgänge zur Gewährleistung der Sicherheit in die Entwicklung der Sicherheitsaspekte des Systems mit aufgenommen werden. Beispielsweise können Dokumente, die vertraulich behandelt werden müssen, ebenso festgehalten werden wie Abläufe zur Beweissicherung.

Die Zugriffs- und Ausführungsrechte für Akteure sind festzulegen. Durch eine hierarchische Anordnung können Gemeinsamkeiten und Besonderheiten in den Rechten frühzeitig modelliert werden. Durch die Spezifikation der Benutzerrechte außerhalb des Klassendiagramms bleibt das Klassendiagramm übersichtlich und Änderungen im Klassendiagramm können ohne Betrachtung von Assoziationen zu Klassen für Benutzerrechte durchgeführt werden. Durch die vorab textuelle Spezifikation der Benutzerrechte müssen diese vorab noch nicht exakt spezifiziert werden und bleiben auch für Endanwender und Auftraggeber verständlich.

Das vorgeschriebene Verfahren bietet eine Vielzahl von Iterationsmöglichkeiten. Es können Teile der Geschäftsprozesse und des Domänenmodells inklusive Betrachtung der Sicherheitsaspekte inkrementell entwickelt werden. Durch die mehrfache Anwendung des Sicherheitsmikroprozesses können auch die Bedrohungen, Risken und Maßnahmen iterativ entwickelt und verfeinert werden.

Durch das vorgestellte Verfahren zur Spezifikation der Sicherheit in den Geschäftsprozessen und im Domänenmodell werden unter Umständen die Aspekte der Zugriffssicherheit in ähnlichen Abläufen mehrfach spezifiziert. Auch für Abläufe auf operierenden Daten als auch für die Daten selbst werden potenzielle Angriffsziele ermittelt. Da jedoch die Sicherheit nur dann gewährleistet werden kann, wenn alle Abläufe und Daten hinreichend betrachtet wurden, ist dieser Overhaed innerhalb der Sicherheitsmodellierung zu tolerieren.

In Beziehung stehende Prozessmuster

Übergeordneter Prozess

- Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme (siehe Abschnitt 5.3.2)

Auszuführender Teilprozess

- Prozessmuster zum Sicherheitsmikroprozess (siehe Abschnitt 5.3.3)

Weitere referenzierte Prozessmuster

- Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme (siehe hierzu Abschnitt 7.5.1) ◇

6.4.2 Anwendung des Sicherheitsmikroprozesses

Innerhalb des Prozesses zur Geschäftsprozessmodellierung zugriffssicherer Systeme wird erstmals der im Prozessmuster 6.3 vorgestellte Sicherheitsmikroprozess angewendet, wobei Bedrohungen, Risiken und Maßnahmen für mit Schutzzielen annotierte Aktivitäten und Objekte des Domänenmodells ermittelt werden. Im Folgenden zeigen wir exemplarisch an unserer *TimeTool*-Fallstudie die Anwendung des Sicherheitsmikroprozesses.

Modellierung von Bedrohungen und Bewertung der Risiken

Im Abschnitt 6.2.1 haben wir im Domänenmodell der *TimeTool*-Fallstudie bereits mit den eingeführten Stereotypen Schutzziele annotiert. In Tabelle 6.1 betrachten wir die Aktivitäten *Modellierung der Bedrohungen* und *Bewertung der Risiken* zu diesen annotierten Objekten des Domänenmodells. Für jedes Objekt, das mindestens mit einem Schutzziel annotiert wurde, erfassen wir die Bedrohungen und schätzen das Risiko ab. Für die Risikoabschätzung verwenden wir nur eine dreistufige Skala (gering, mittel und hoch). Diese Risikoabschätzung kann auch durch umfangreichere Ermittlungsverfahren ersetzt werden, wie beispielsweise in [HB03]. Auf Verfahren zur Risikoabschätzung gehen wir hier nicht näher ein.

Da Bedrohungen und Risiken innerhalb der weiteren Bearbeitungsschritte, wie etwa zur Ermittlung von Gegenmaßnahmen, referenziert werden, nummerieren wir die Bedrohungen und Risiken durch.

Tabelle 6.1: Bedrohungen und Risiken zum Domänenmodell der TimeTool-Fallstudie

Klasse		Bedrohung und Risiko
Projekt	B001	Ein Projektmitarbeiter oder ein weiterer Benutzer des Systems verändert das Budget des Projekts.
	R001	Das Risiko wird als *gering* eingeschätzt, da das Budget in der Klasse Projekt nur zu Informationszwecken gespeichert wird und keinen weiteren Einfluss auf das System hat.
Activity	B002	Ein Projektmitarbeiter oder ein weiterer Benutzer des Systems verändert die Dauer und Start- bzw. Endzeiten von Projektaktivitäten.
	R002	Das Risiko wird als *hoch* eingeschätzt, da durch diese Veränderung Mitarbeiter zusätzliche Stunden auf ein Projekt buchen können. Zudem wird der Projektplan inkonsistent.
TeamWorker	B003	Ein Projektmitarbeiter, ein Projektmanager oder ein weiterer Benutzer des Systems versucht, das Passwort eines Projektmitarbeiters zu lesen oder zu verändern.
	R003	Das Risiko wird als *hoch* eingeschätzt, da dadurch falsche Stundenbuchungen eingetragen werden können.
	B004	Ein Projektmitarbeiter, ein Projektleiter oder ein weiterer Benutzer des Systems liest und verändert private Daten eines Projektmitarbeiters.

Fortsetzung auf der nächsten Seite

Fortsetzung von Tabelle 6.1

	R004	Das Risiko wird als *mittel* eingestuft, da das lesen privater Attribute, wie etwa der Adresse, Telefonnummer, etc., dem Datenschutz widerspricht. Beispielsweise können diese Daten für Mobbing verwendet werden. Weiterhin kann durch die Abänderung z.B. der Telefonnummer der Projektmitarbeiter im Notfall nicht erreicht werden.
ProjektManager	B005	Ein Projektmitarbeiter, ein Projektmanager oder ein weiterer Benutzer des Systems versucht, das Passwort eines Projektmitarbeiters zu lesen oder zu verändern.
	R005	Das Risiko wird als *hoch* eingeschätzt, da dadurch Aktionen, die nur vom Projektleiter ausgeführt werden dürfen, durch einen Projektmitarbeiter, eines Projektleiters eines weiteren Projekts oder durch einen weiteren Benutzer manipuliert werden können.
	B006	Ein Projektmitarbeiter, ein Projektleiter oder ein weiterer Benutzer des Systems liest und verändert private Daten eines Projektmitarbeiters.
	R006	Das Risiko wird als *hoch* eingestuft, da das lesen privater Attribute, wie z.B. der Adresse oder Telefonnummer, dem Datenschutz widerspricht. Z.B. können diese Daten für Mobbing verwendet werden. Weiterhin kann durch die Abänderung etwa der Telefonnummer der Projektleiter im Notfall nicht erreicht werden.
Administrator	B007	Ein Projektmitarbeiter, ein Projektmanager oder ein weiterer Benutzer des Systems versucht, das Passwort eines Administrators zu lesen oder zu verändern.
	R007	Das Risiko wird als *hoch* eingeschätzt, da mit der Administratorberechtigung im System alle Daten projektübergreifend gelesen oder manipuliert werden können.
	B008	Ein Projektmitarbeiter, ein Projektleiter oder ein weiterer Benutzer des Systems liest und verändert private Daten eines Administrators.
	R008	Das Risiko wird als *hoch* eingestuft, da das lesen privater Attribute, wie der Adresse, Telefonnummer, etc., dem Datenschutz widerspricht. Beispielsweise können diese Daten für Mobbing verwendet werden. Weiterhin kann durch die Abänderung zum Beispiel der Telefonnummer der Administrator im Notfall nicht erreicht werden.
Accounting	B009	Ein Benutzer des Systems versucht, fremde Buchungen zu lesen.
	R009	Das Risiko wird als *mittel* eingestuft, da dies zu einer Überwachung von Projektmitarbeitern oder Mitarbeitern aus anderen Projekten führen kann. Weiterhin können aus den Buchungsdaten von unbefugten Personen Statistiken über das Projekt angefertigt werden.
	B010	Benutzer des Systems versuchen, fremde oder eigene Buchungen zu manipulieren.
	R010	Das Risiko wird als *hoch* eingestuft, da dadurch Kollegen aus eigenen oder fremden Projekten geschädigt werden können sowie die Zeit- und Budgetplanung für Projekte indirekt manipuliert wird. Projektmitarbeiter könnten dadurch zusätzlich Stunden auf bereits überprüfte Aktivitäten zu buchen.

Fortsetzung auf der nächsten Seite

Fortsetzung von Tabelle 6.1

	B011	Der Projektmitarbeiter gibt eine Korrekturbuchung für eine Projektaktivität nach der Kontrolle durch den Projektmanager ein.
	R011	Das Risiko wird als *hoch* eingestuft, da der Projektmitarbeiter durch diese Korrekturbuchungen zu einer Projektaktivität zu viel geleistete Stunden verbuchen kann.
	B012	Ein Projektmanager gibt Stundenbuchungen mit negativen Stunden ein.
	R012	Das Risiko wird als *hoch* eingestuft, da der Projektmanager durch negative Stundenbuchungen Budgetüberschreitungen vertuschen kann.
	B013	Ein Projektmitarbeiter bestreitet, dass er an einer Stundenbuchung, für die er die Lese- und Schreibberechtungen innehat, Änderungen durchgeführt hat.
	R013	Das Risiko wird als *hoch* eingestuft, denn durch fehlerhafte Stundenbuchungen werden die Projektkosten falsch berechnet. Weiterhin treten Inkonsistenzen bei der Überprüfung des Zeitmanagements auf.

Neben den Domänenobjekten werden auch innerhalb der Geschäftsprozesse sicherheitskritische Aktivitäten mit den in Abschnitt 6.2 eingeführten Schutzziel-Stereotypen annotiert. Im Folgenden erweitern wir das Bedrohungs- und Risikomodell. Wir ermitteln für den in Abbildung 6.13 dargestellten Geschäftsprozess Bedrohungen und Risiken analog zu denen des Domänenmodells.

Tabelle 6.2: Bedrohungen und Risiken zum TimeTool-Geschäftsprozess aus Abbildung 6.13

Aktivität		**Bedrohung und Risiko**
initialize project	B014	Ein Projektmitarbeiter, ein Projektleiter aus einem anderen Projekt oder ein weiterer Nutzer des Systems manipulieren die zu Projektbeginn erstellten Projektaktivitäten, Projektmitarbeiter sowie die Projektinformation.
	R014	Das Risiko wird als *mittel* eingeschätzt, da sich Inkonsistenzen im geplanten Projektablauf und in den Budgetplanungen ergeben. Unzulässig zugeordnete Projektmitarbeiter haben somit Zugang zum Projekt und können legitime Änderungen und Buchungen ausführen sowie vertrauliche Projektinformationen lesen.
change project data	B015	Daten, die während der Initialisierung und während Stundenbuchungen im System eingetragen wurden, können in dieser Aktivität verändert werden. Projektadministratoren, Projektmitarbeiter sowie weitere Nutzer des Systems können sowohl Änderungen an der Projektinitialisierung als auch an den Stundenbuchungen durchführen.
	R015	Das Risiko wird *hoch* eingestuft, da durch unbefugte Änderungen an den Stundenbuchungen falsche Budgetberechnungen ergeben können. Weiterhin können sich, wie in der Risikobewertung R012 bereits beschrieben wurde, Inkonsistenzen im Projektablauf ergeben und Änderungen in Buchungen können legitim werden, indem die Projektaktivitäten angepasst werden.

Fortsetzung auf der nächsten Seite

Fortsetzung von Tabelle 6.2

account online	B016	Beim Buchen von Stunden innerhalb des Systems, d.h., beim Online-Buchen, können die Daten von unbefugten Akteuren des Systems gelesen bzw. modifiziert werden.
	R016	Das Risiko wird *hoch* eingestuft, da vertrauliche Projektinformation zur Erstellung von Statistiken benutzt werden kann und somit für ein Unternehmen sicherheitskritische Projektinformationen erlangt werden können. Die Abänderung von Buchungsdaten kann zudem zu einer Überschreitung des Budgets führen.
	B017	Projektmitarbeiter, deren Stundenberechnungen bei der Kontrolle durch den Projektleiter fehlerhaft sind, behaupten, dass sie keine falschen Stundenbuchungen (z.B. zu viele Stunden gebucht) durchgeführt haben. Weiterhin können sie etwa behaupten, dass sie fehlende Stundenbuchen eingetragen haben, obwohl sie dies in Wirklichkeit nicht durchgeführt haben.
	R017	Das Risiko wird als *hoch* eingeschätzt, da durch fehlerhafte Stundenbuchungen Budgetüberschreiten auftreten und Terminprobleme nicht rechtzeitig erkannt werden können.
account offline	B018	Buchungsdaten, die auf einem externen Rechner eingegeben werden, sind für Angriffe besonders anfällig. Auf dem Offline-Rechner können Attacken zum Auslesen und Modifizieren der zwischengespeicherten Buchungsdaten ausgeführt werden. Angreifer können entweder weiter Projektmitarbeiter, Projektleiter anderer Projekte oder weitere Systemnutzer sein.
	R018	Das Risiko wird als *hoch* eingeschätzt, da sich durch unbefugte Änderungen an den Stundenbuchungen fehlerhafte Budgetberechnungen ergeben. Weiterhin können durch derartige Angriffe auch Projektmitarbeiter von Teamkollegen, etc. unzulässig überwacht werden. Eine besondere Gefahr bei Offline-Rechnern besteht darin, dass die Buchungsdaten auf diesen Rechnern zwischengespeichert werden müssen und diese Daten z.B. über das Filesystem oder eine Datenbank ausgelesen und modifiziert werden können. Dies wird u.a. dadurch unterstützt, dass derartige Offline-Rechner (wie zum Beispiel Notebooks) in privaten Umgebungen ohne ausreichendem Schutz (aktueller Virenscanner, OS-Update, Firewall) über Modemverbindungen mit dem Internet verbunden sind.
prepare monthly report	B019	Unbefugte Akteure können auf Statistiken und Berichte Angriffe starten, die bei der Erstellung des monatlichen Berichts auftreten. Sie versuchen dabei, auf vertrauliche Information zuzugreifen und diese zu manipulieren.
	R019	Das Risiko wird als *hoch* eingestuft, da geheime Projektinformationen ermittelt und Statistiken gefälscht werden können. Insbesondere können Projektbeteiligte konkurrierender Projekte versuchen, das gegnerische Projekt negativ darzustellen.

Die Liste der Bedrohungen mit ihren Risiken ist für alle Geschäftsprozesse zu ergänzen. Wurden in den Geschäftsprozessen auch Schutzziele zu Flussobjekten und zur Autorisierung ermittelt (vgl. Abbildungen 6.16 und 6.17), so sind für diese potenziellen Angriffsziele ebenfalls Bedrohungen und Risiken zu ermitteln.

Da Schutzziele zur Autorisierung, wie sie in Abbildung 6.15 eingeführt wurden, keiner Klasse oder Aktivität zugeordnet sind, betrachten wir die potenziellen Angriffe gegen die Autorisierung bei der Modellierung der Bedrohungen und Risiken des zugehörigen Flussobjektes.

Entwurf von Maßnahmen

Nachdem die Bedrohungen und Risiken ermittelt wurden, sind Maßnahmen zu entwickeln, die den Bedrohungen entgegenwirken, soweit das Risiko nicht als minder gering eingestuft werden kann.

Nicht alle Bedrohungen können bereits in der Phase der Geschäftsprozessmodellierung zugriffssicherer Systeme behandelt werden, da zu diesem Zeitpunkt die internen Abläufe oftmals noch nicht geklärt sind oder Protokolle noch nicht ausgewählt werden können. Es gibt jedoch eine Reihe von Bedrohungen, denen bereits durch Änderungen in den Geschäftsprozessen oder im Domänenmodell entgegengewirkt werden kann. Diejenigen Bedrohungen, die zu diesem Zeitpunkt noch nicht behandelt werden können, sind entweder im Rahmen der Anwendungsfallmodellierung oder der Analyse zugriffssicherer Systeme zu behandeln. In jedem dieser Prozessabschnitte wird der Sicherheitsmikroprozess ausgeführt (vgl. Abbildung 5.7), sodass dann unter den hinzugewonnenen Erkenntnissen geeignete Maßnahmen ausgewählt werden können.

Im Folgenden zeigen wir an Bedrohungen zur Klasse *Accounting* (vgl. Tabelle 6.1), welche Bedrohungen bereits innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme betrachtet werden können. Dafür können wir folgende Maßnahmen ableiten:

- Da die Bedrohungen B009 bis B013 alle mit dem Risiko *mittel* oder *hoch* eingestuft wurden, müssen für all diese Bedrohungen Maßnahmen entwickelt werden.

- Für die Bedrohungen B009 und B010 sind die Daten verschlüsselt im System abzulegen. Protokolle zur Datenverschlüsselung und Auswirkungen auf das Domänenmodell sowie auf die Prozessabläufe werden im Rahmen der Anwendungsfallmodellierung und der Analyse zugriffssicherer Systeme näher betrachtet.
  Projektmitarbeiter dürfen nur eigene Stundenbuchungen lesen und ändern, Projektleiter nur Buchungen zu eigenen Projekten. Dies ist bei der Modellierung der Benutzerrechte zu berücksichtigen.

- Für die Bedrohung B011 ergibt sich folgender Projektablauf: Nachdem eine Projektaktivität beendet wurde und alle Projektmitarbeiter die Stundenbuchungen eingetragen haben, kontrolliert der Projektleiter die Stundenbuchungen. Anschließend setzt er in der zugehörigen Projektaktivität den Status der Projektaktivität auf *frozen* und es können an den zugehörigen Buchungen keine Änderungen mehr durchgeführt werden. Somit ist im Domänenmodell zur Klasse *Activty* das Statusattribut *State* mit den möglichen Belegungen *frozen, released* hinzuzufügen. Bei der Spezifikation der Benutzerrechte ist dieses Statusattribut einzubeziehen. Es können Buchungen nur dann verändert werden, wenn der Status der zugehörigen Aktivität auf *released* gesetzt wurde.

- Der Bedrohung B012 kann entgegengewirkt werden, indem negative Stundenbuchungen grundsätzlich verboten werden. Diese Bedingung wird in den Regeln zum Geschäftsprozessmodell aufgenommen.

- Damit Projektmitarbeiter Änderungen und Eintragungen zu Stundenbuchungen nicht abstreiten können, werden alle Eintragungen und Änderungen in einer so genannten Historie mitprotokolliert. Als Maßnahme zur Bedrohung B013 fügen wir deshalb zum Domänenmodell eine Klasse *Logging* hinzu, die alle Änderungen an Buchungseinträgen speichert. Der Zugriff auf die Klasse *Logging* muss bei der Ermittlung der Benutzerrechte betrachtet werden. Die Abläufe, bei denen Einträge zu Stundenbuchungen erzeugt oder verändert werden, sind entsprechend anzupassen.

Aus den Bedrohungen B011 und B013 ergeben sich Änderungen am Domänenmodell. In Abbildung 6.32 sind diese Änderungen am Klassendiagramm zum Domänenmodell grau gekennzeichnet.

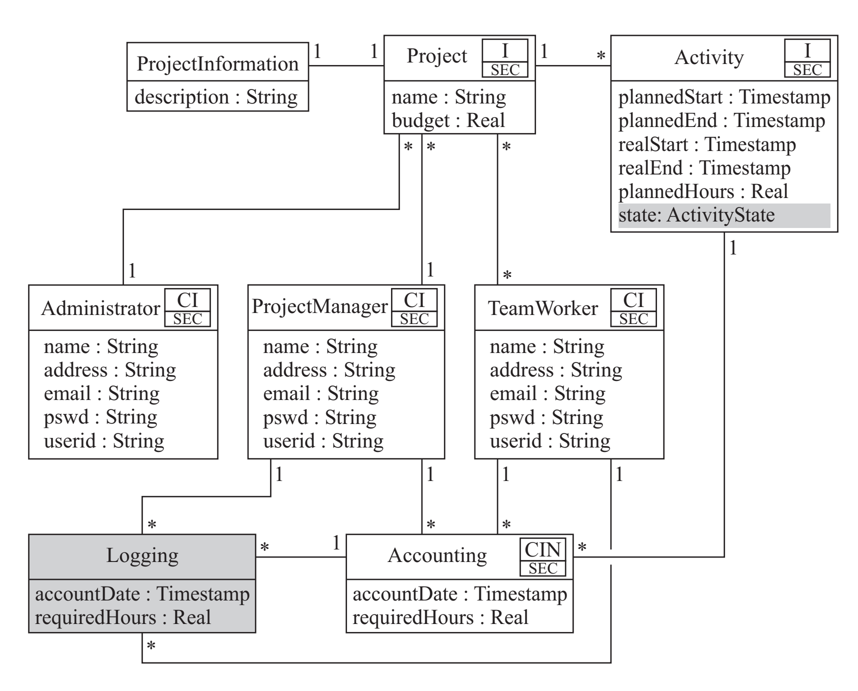


Abbildung 6.32: Ergänzungen am TimeTool-Domänenmodell zur Abwehr von Bedrohungen

Alle Änderungen müssen ebenfalls einer Analyse der Sicherheitsaspekte unterzogen werden. An den Klassen *Logging* und *Activity* muss nun entweder im Rahmen der Geschäftsprozessmodellierung zugriffssicherer Systeme oder der Anwendungsfallmodellierung zugriffssicherer Systeme der Sicherheitsmikroprozess ausgeführt werden und die Benutzerrechte sind zu spezifizieren.

Wir führen hier für die eingeführten und geänderten Klassen den Sicherheitsmikroprozess nicht nochmals in der Geschäftsprozessmodellierung zugriffssicherer Systeme aus, sondern befassen uns mit diesen Änderungen im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme.

Überprüfung der Maßnahmen

Nach dem Entwurf der Abwehrmaßnahmen verlangt der Sicherheitsmikroprozess eine Überprüfung der eingeführten Maßnahmen. Dabei ist informal, semiformal oder formal zu über-

prüfen, inwiefern die eingeführten Maßnahmen zum Ziel führen, d.h., den ermittelten Bedrohungen entgegenwirken.

Im Rahmen unserer Fallstudie können wir die Maßnahmen informal überprüfen. Die eingeführte Klasse *Logging* speichert alle Änderungen an Buchungen. Unter der Voraussetzung, dass in der weiteren Modellierung alle Änderungen darin mitprotokolliert werden und dass Benutzer des Systems die Logging-Informationen nicht verändern können, wird dadurch ein Kontrollinstrument eingeführt, sodass die Verbindlichkeit von Buchungen und Buchungsänderungen gewährleistet wird.

Ebenso wirkt die Maßnahme des zusätzlichen Statusattributs innerhalb der Klasse *Activity* einer anschließenden Manipulation entgegen. Voraussetzung ist hier ebenfalls, dass Benutzer diesen Status nicht ändern können, was bei der weiteren Modellierung der Abläufe und Benutzerrechte zu beachten ist.

6.5 Produktartefakte der Geschäftsprozessmodellierung zugriffssicherer Systeme

Im Abschnitt 6.1 wurden die Produktartefakte der allgemeinen Geschäftsprozessmodellierung vorgestellt. Im Rahmen der Geschäftsprozessmodellierung zugriffssicherer Systeme werden diese Produktartefakte mit Aspekten der Zugriffssicherheit angereichert und zusätzliche Produktartefakte, so genannte *Sicherheitsdokumente*, werden hinzugefügt.

Beziehungen zur Erweiterung der Produktartefakte

Allgemein sprechen wir hier von einer Erweiterung der Produktartefakte. Bei einer genaueren Untersuchung der Beziehungen lässt sich diese Erweiterung genauer klassifizieren.

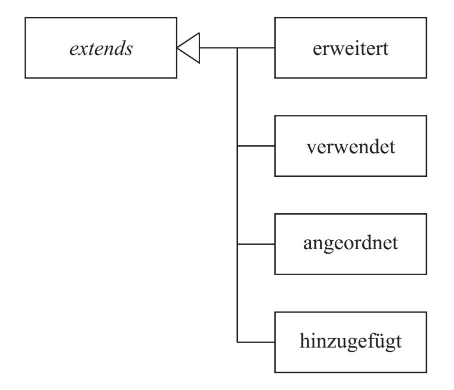


Abbildung 6.33: Beziehungen bei der Erweiterung der Produktartefakte

Wir können die in Abbildung 6.33 gezeigten vier Beziehungstypen zwischen Produktartefakten der allgemeinen Modellierung und der Modellierung zugriffssicherer Systeme als Verfeinerung der allgemeinen *extends*-Beziehung feststellen. Die verfeinerten Beziehungen haben folgende Bedeutung:

erweitert: Produktartefakte der allgemeinen Modellierung werden erweitert. Diese Erweiterung umfasst folgende Änderungen an Standardprodukten:

- Klassendiagramme und Aktivitätsdiagramme werden um Klassen und Aktivitäten erweitert.
- Es werden vollständig neue Aktivitätsdiagramme zur Beschreibung von Abläufen hinzugefügt.
- In UML-Diagrammen werden durch den Erweiterungsmechanismus der UML (siehe [OMG03, BRJ98, EP00]) *Stereotypen*, *Constraints* oder *Tagged Values* zur Beschreibung von Sicherheitsanforderungen eingefügt.
- Textuelle Beschreibungen werden ergänzt.

verwendet: Produktartefakte der allgemeinen Modellierung werden innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme verwendet, aber nicht verändert.

angeordnet: Ermittelte Entitäten, wie beispielsweise Akteure oder Klassen, werden hierarchisch angeordnet. Im Rahmen der Spezifikation der Benutzerrechte spielt diese hierarchische Anordnung der Akteure und Klassen eine bedeutende Rolle.

hinzugefügt: Neue Produktartefakte zur Beschreibung von Sicherheitsaspekten werden zu den herkömmlichen Produktartefakten hinzugefügt.

Erweiterung der Produktartefakte

Im Prozessmuster zur Geschäftsprozessmodellierung zugriffssicherer Systeme wurden die Produktartefakte bereits vorgestellt. Wir betrachten im Folgenden die Produktartefakte zur Geschäftsprozessmodellierung zugriffssicherer Systeme sowie die verfeinerten Beziehungen in der vorab vorgestellten Art.

Abbildung 6.34 zeigt die Produktartefakte der Geschäftsprozessmodellierung zugriffssicherer Systeme mit ihren Teilprodukten. In der Abbildung sind alle durch Sicherheitsaspekte veränderten oder hinzugefügten Teilprodukte grau hinterlegt. In den Teilprodukten ist zudem auch die Beziehung der Produktartefakte zu den Produktartefakten der allgemeinen Geschäftsprozessmodellierung gekennzeichnet.

Die Teilprodukte des *Business Vision Modells* werden verwendet. Bezüglich der Modellierung der Zugriffssicherheit werden diese Teilprodukte nicht verändert.

Die Geschäftsobjekte als Teil des *Domänenmodells* werden erweitert. Klassen des Domänenmodells werden mit Attributen erweitert, die sich aus Maßnahmen zur Abdeckung von Bedrohungen ergeben. Weiterhin können derartige Maßnahmen auch dazu führen, dass neue Klassen ins Domänenmodell aufgenommen werden. Die Klassen des Domänenmodells werden mit Schutzzielen unter der Verwendung von Schutzziel-Stereotypen (vgl. Abschnitt 6.2.1) annotiert.

Im Rahmen der Geschäftsprozessmodellierung zugriffssicherer Systeme werden innerhalb der *Geschäftsprozessmodellierung* die Prozesse ebenfalls mit Schutzziel-Stereotypen annotiert und die Abläufe sind an die ermittelten Maßnahmen zur Abdeckung der Bedrohungen anzupassen. Die ermittelten Akteure sind in Hinblick auf eine Spezifikation der Benutzerrechte hierarchisch anzuordnen. Im Glossar werden Begriffe hinterlegt, die bei der gemeinsamen Erarbeitung von Auftraggebern, Auftragnehmern, Anforderungsentwicklern und Endanwendern zu Beginn der Geschäftsprozessmodellierung zu definieren sind. Allgemeine Regeln des Geschäftsprozessmodells, Ergänzungen zur Systembeschreibung und die Architektur sind bei der Erarbeitung der Sicherheitsaspekte mit heranzuziehen, aber nicht zu ändern.

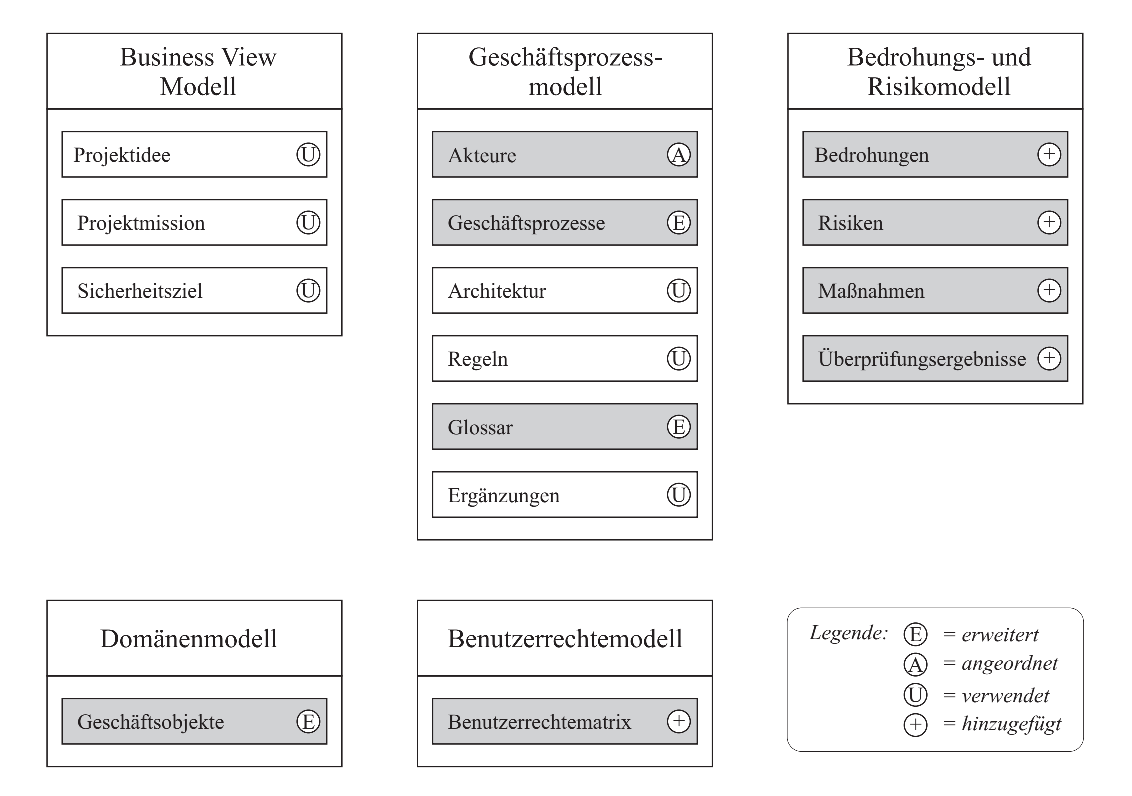


Abbildung 6.34: Produktartefakte der Geschäftsprozessmodellierung zugriffssicherer Systeme

Das *Bedrohungs- und Risikomodell* ist ein eigenes Produktartefakt der Modellierung zugriffssicherer Systeme. Dieses ist in der gewöhnlichen Geschäftsprozessmodellierung nicht vorhanden und wird deshalb zur Systembeschreibung hinzugefügt. Bedrohungen und Risiken werden textuell in Auflistungen oder Tabellen beschrieben. Für eine Referenzierung sind diese zu nummerieren. Maßnahmen können ebenfalls textuell oder semiformal beispielsweise mithilfe von UML-Diagrammen beschrieben werden. Die Eignung der Maßnahmen kann textuell, semiformal oder formal aufgezeigt werden.

Ein weiteres eigenes Produktartefakt der Modellierung zugriffssicherer Systeme stellt das *Benutzerrechtemodell* dar. Hier werden die Zugriffs- und Ausführungsrechte tabellarisch textuell und prädikativ beschrieben.

6.6 Zusammenfassung

Die mehrstufige, durchgängige und sukzessive Analyse von Aspekten der Zugriffssicherheit beginnt im vorgestellten Ansatz bereits mit der Modellierung der Geschäftsprozesse. Die Modellierung wird unabhängig von technischen Details auf der Abstraktionsebene der Abläufe und des Domänenmodells begonnen. In Hinblick auf eine Modellierung der Zugriffsrechte und der Bedrohungen werden hierzu potenzielle Angriffsmöglichkeiten innerhalb der Geschäftsabläufe und des Klassendiagramms ermittelt und mit Schutzzielen annotiert.

Für die Annotation von Schutzzielen sind adäquate Beschreibungstechniken notwendig. Für die Modellierung der Abläufe verwenden wir die von der UML zur Verfügung stehenden

Diagrammtypen Klassendiagramm und Aktivitätsdiagramm. Da jedoch die UML innerhalb dieser Diagrammtypen keine Konstrukte zur Annotation von Schutzzielen vorsieht, erweitern wir die UML um die geforderten Konstrukte. Mithilfe des Erweiterungsmechanismus der UML führen wir Stereotypen für die Annotation von Schutzzielen in Klassendiagrammen und Aktivitätsdiagrammen ein.

Mit dem eingeführten Prozess zur Geschäftsprozessmodellierung zugriffssicherer Systeme wird weiterhin das Ziel verfolgt, dass die Anforderungen an die Sicherheit und Systemfunktionalität gemeinsam entwickelt werden. Bei der lokalen Betrachtung eines Ablaufs oder eines Ausschnitts aus dem Domänenmodell können so die Sicherheitsaspekte losgelöst von weiteren Abläufen und Daten auf einer hohen Abstraktionsebene spezifiziert werden.

Eine iterative Entwicklung wird insbesondere durch die Anwendung des Sicherheitsmikroprozesses und durch die Iterationsmöglichkeiten innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme erreicht. Benutzerrechte, Bedrohungen, Risiken und Gegenmaßnahmen können nach der Ermittlung aller Schutzziele innerhalb der Abläufe und Daten modelliert werden. Genauso können diese aber auch bereits nach der Ausarbeitung einer Kernfunktionalität ermittelt und anschließend schrittweise erweitert werden.

Der entwickelte Ablauf zur Geschäftsprozessmodellierung zugriffssicherer Systeme wurde in einem Prozessmuster formuliert und die in die Abarbeitung zentralen Produktartefakte wurden vorgesellt. Zur Spezifikation der Ergebnisse sind die Produktartefakte der Geschäftsprozessmodellierung zu erweitern und um neue zu ergänzen.

Im Rahmen der folgenden Anwendungsfallmodellierung zugriffssicherer Systeme werden die bereits erarbeiteten Produkte verfeinert und ergänzt. Insbesondere sind die zu automatisierenden Abläufe in Anwendungsfälle überzuführen, die Benutzerrechte hinsichtlich einer Generierung zu untersuchen und die Authentifikation als Systemfunktionalität zu integrieren.

7 Die Anwendungsfallmodellierung zugriffssicherer Systeme

Im Rahmen der im vorausgehenden Kapitel eingeführten Geschäftsprozessmodellierung zugriffssicherer Systeme wurden betriebliche Prozesse und eine erste Version des Domänenmodells des zu entwickelnden Systems festgelegt. Schutzziele und Benutzerrechte auf der gewonnenen Struktur und dem ermittelten Verhalten wurden spezifiziert. Dieses Kapitel beschäftigt sich mit den darauf aufbauenden Aufgaben des zweiten Prozessabschnitts der Anforderungsspezifikation zugriffssicherer Systeme, die im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme auszuführen sind.

Bei der Anwendungsfallmodellierung werden aus den innerhalb der Geschäftsprozessmodellierung gewonnenen Abläufen diejenigen bestimmt, die innerhalb des Systems umgesetzt werden. Die Aktivitäten als einzelne Bausteine der Abläufe werden ihren ausführenden Akteuren zugeordnet und in so genannten Anwendungsfällen strukturell beschrieben. Anwendungsfalldiagramme geben eine Übersicht über die Zusammenhänge zwischen ausführenden Akteuren und untereinander agierenden Anwendungsfällen. Das Domänenmodell aus der Geschäftsprozessmodellierung muss angepasst werden. Insbesondere sind nur diejenigen Entitäten des Domänenmodells weiterhin zu betrachten, die bei den umzusetzenden Abläufen zur Datenhaltung notwendig sind. Im Klassendiagramm sind Redundanzen zu eliminieren, Gemeinsamkeiten zu erkennen und Assoziationen anzupassen. Ein Überblick über die Aktivitäten der Anwendungsfallmodellierung gibt Abschnitt 7.1.

Ebenso wie bei der heute üblichen Geschäftsprozessmodellierung werden Sicherheitsaspekte bei der Anwendungsfallmodellierung kaum betrachtet. Auch hier steht die Entwicklung der Funktionalität im Vordergrund. Nach [JBR99] werden nichtfunktionale Anforderungen, unter die auch Sicherheitsanforderungen eingeordnet werden können, innerhalb der Spezifikation der Anwendungsfälle nur aufgenommen, die Ausarbeitung aber in die Designphase verschoben.

Innerhalb der Anwendungsfallmodellierung kann jedoch die in der Geschäftsprozessmodellierung zugriffssicherer Systeme begonnene Sicherheitsanalyse fortgesetzt werden. Für definierte Bedrohungen müssen die ermittelten Maßnahmen in die Abläufe integriert werden. Neu hinzugefügte oder modifizierte Abläufe sowie Entitäten des Domänenmodells können iterativ nach dem bereits vorgestellten Verfahren (siehe Kapitel 6) der Schutzzielanalyse mit anschließender Ermittlung der Bedrohungen, Risiken und Maßnahmen unterzogen werden. Die Benutzerrechte auf Aktivitäten und Entitäten sind anzupassen, zu erweitern und gegebenenfalls zu konkretisieren. Das Schutzziel Authentifikation, das bis jetzt nur eine untergeordnete Rolle gespielt hat, ist im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme detailliert zu betrachten.

Zur Darstellung des Schutzziels Authentifikation erweitern wir in Abschnitt 7.2 Aktivitätsdiagramme. Eine Modellierung der Autorisierung erfordert zudem die Beschreibung von Sitzungen in UML-Aktivitätsdiagammen. Innerhalb der Verfeinerung von Authentifikation und

Sitzungen betrachten wir die Trennung zwischen Authentifikation und Autorisierung sowie den Übergang von der exemplarischen Authentifikation zur allgemeinen Authentifikation, die zumeist nur zu Beginn einer Systemnutzung ausgeführt wird.

Sicherheitsanforderungen können die Abläufe innerhalb der Anwendungsfälle verändern oder eigenständige, zusätzliche Anwendungsfälle erfordern. In Abschnitt 7.3 betrachten wir die Entwicklung von Sicherheitsanwendungsfällen und deren strukturierte Beschreibung.

Modifikationen an den Abläufen, Akteuren und am Domänenmodell erfordern auch eine Anpassung der bereits spezifizierten Benutzerrechte. Abschnitt 7.4 beschreibt die Evolution der Berechtigungen und geht dabei auch auf die Zusammenhänge der Benutzerrechte in Struktur und Verhalten genauer ein. Für Abläufe, in denen zum Zeitpunkt der Spezifikation der Anwendungsfälle schon klar definiert werden kann, auf welche Teilabläufe und Daten zugegriffen wird, können die Ausführungsrechte des Anwendungsfalls aus den benötigten Benutzerrechten im Ablauf errechnet werden.

Abschnitt 7.5 stellt das Vorgehen in der Anwendungsfallmodellierung zugriffssicherer Systeme in Form eines Prozessmusters vor. Dabei kommt erneut der Sicherheitsmikroprozess aus Abschnitt 5.3.3 zur Anwendung. Auf die Produktartefakte der Anwendungsfallmodellierung zugriffssicherer Systeme gehen wir in abschließend Abschnitt 7.6 ein.

7.1 Grundlagen der Anwendungsfallmodellierung

Bevor wir auf die Besonderheiten der Anwendungsfallmodellierung zugriffssicherer Systeme eingehen, geben wir einen kurzen Überblick über Ziele, Aktivitäten und Produktartefakte der allgemeinen Anwendungsfallmodellierung ein.

7.1.1 Motivation

Mit der Entwicklung neuerer Entwurfsmethoden wurde der reine Entwurf von Objekten und Operationen verlassen. Es wurden neue Modellierungskonzepte, wie beispielsweise Systemoperationen [CAB+94], Geschäftsvorfälle [Den91] und Anwendungsfälle [JCJO92, JBR99] eingeführt, bei denen in den frühen Phasen des Entwurfs zur Modellierung von Struktur und Verhalten ein hybrides Vorgehen unterstützt wird. Bei dem prominentesten Ansatz zu den Anwendungsfällen (engl. *Use Cases*) wird sowohl eine objektorientierte als auch eine funktionsorientierte Sicht auf das System eingenommen, in der sowohl die Objekte des Anwendungskerns (Domänenmodell) als auch die Dienste, die das System unterstützt, beschrieben werden (vgl. [Bre01]). Die weitere Entwicklung ist durch die Integration der Anwendungsfälle in das objektorientierte Modell getrieben.

In der hybriden Sichtweise werden Klassendiagramme zur Konzeption des Anwendungskerns sowie Anwendungsfalldiagramme zusammen mit schematischen textuellen Beschreibungen als Beschreibungstechniken eingesetzt. Die objektorientierte und funktionsorientierte Sicht sind dabei lose gekoppelt und können zeitgleich erstellt werden.

Diese hybride Sichtweise des Systementwurfs eignet sich zur Fortsetzung unseres bereits vorgestellten Entwurfs der Geschäftsprozessmodellierung zugriffssicherer Systeme. Vorarbeiten

im Bereich der Geschäftsprozesse werden bei der Beschreibung des Verhaltens wieder verwendet und in Anwendungsfalldiagramme und textuelle Beschreibungen überführt. Das bereits erstellte Domänenmodell ist zu übernehmen, zu verfeinern und zu ergänzen.

Bei der Wiederverwendung der Ergebnisse aus der Geschäftsprozessmodellierung ist insbesondere zu beachten, dass bei der Beschreibung der Geschäftsprozesse und des Domänenmodells auch Bestandteile modelliert worden sein können, die nicht automatisiert werden, d.h., die nicht im zu entwickelnden System umgesetzt werden. Aus diesem Grund sind bei der Überführung der Abläufe diese zunächst anzupassen, d.h., manuelle Aktivitäten sind in den Abläufen zu entfernen und das Domänenmodell darf nur Klassen enthalten, die in dem automatisierten Ablauf verwendet werden. Entsprechend ist auch das Akteurmodell anzupassen, denn durch die Selektion der automatisierbaren Abläufe können durchaus Akteure entfernt werden.

Anwendungsfälle

Anwendungsfälle sind nach [BRJ98, Bre01] Dienste, die das System einem externen Akteur anbietet. Externe Akteure repräsentieren dabei alle Entitäten, die mit dem System interagieren. Ein Anwendungsfall ist eine Abfolge von Aufgaben, die durch ein System ausgeführt werden, und die ein Ergebnis von messbarem Wert für einen bestimmten Akteur hervorbringen. Die Erledigung der Aufgabe erfolgt in Abhängigkeit von den Daten des Systems, Daten können dabei gelesen und verändert werden. Unter *System* verstehen wir in diesem Kontext das zu entwerfende Softwaresystem.

Akteure können in diesem Zusammenhang einzelne Personen, eine Maschine, ein anderes Softwaresystem oder ein Dienst [DGP+04b, DGP+04a] sein. Wie im vorgestellten Ansatz zum akteurzentrierten Modell (vgl. Kapitel 4) bereits erklärt wurde, ist bei einem Akteur nicht die einzelne Person an sich entscheidend, sondern die Rolle, die sie im Anwendungsfall einnimmt. Personen können mehrere, unterschiedliche Rollen einnehmen. Anwendungsfälle beschreiben Interaktionen von einem oder mehreren Akteuren mit dem System und die Akteure können sowohl Quelle als auch Senke der Information sein. In unserer *TimeTool*-Fallstudie können beispielsweise die beiden Rollen *Team Worker* und *Projektleiter* von derselben Person eingenommen werden. Ein weiterer Akteur ist der *Web-Browser*.

Ein Anwendungsfall stellt immer einen zusammenhängenden und vollständigen Ablauf dar, sodass ein Anwender eine Aufgabe in Interaktion mit dem System ohne Unterbrechung erledigen kann. Die unterbrechungsfreie Ausführung ist jedoch nicht zwingend gefordert, da beispielsweise ein Akteur diese eigenständig unterbrechen kann, wenn er beispielsweise mehrere Aufgaben gleichzeitig bearbeitet. Durch diese Definition werden jedoch Aufgaben ausgeschlossen, die nicht rechnergestützt durchgeführt werden können, wie beispielsweise die manuelle Übergabe von Dokumenten.

Alle Anwendungsfälle eines zu entwickelnden Systems zusammen beschreiben die Gesamtfunktionalität des Systems. Alle zur Beschreibung eines Systems erstellten Anwendungsfälle werden als Anwendungsfallmodell bezeichnet. Zur Modellierung des vollständigen Systemumfangs und der Modellierungsgrenzen können die Anwendungsfälle des Anwendungsfallmodells in Anwendungsfalldiagrammen dargestellt werden. Ein Anwendungsfalldiagramm stellt somit die Zusammenhänge zwischen Anwendungsfällen untereinander und den Akteuren eines Systems dar. Ein Anwendungsfalldiagramm zur *TimeTool*-Fallstudie ist in Abbildung 3.2 dargestellt.

7.1.2 Aktivitäten der Anwendungsfallmodellierung

Im Rahmen der Geschäftsprozessmodellierung wurden die Abläufe und eine erste Version des Domänenmodells erstellt. Da jedoch zu diesem Zeitpunkt die Grenzen des Systems noch nicht bestimmt waren, sind zu Beginn der Anwendungsfallmodellierung die Grenzen und die Funktionalität des zu entwickelnden Systems festzulegen. Für die spezifizierten Geschäftsabläufe ist festzulegen, welche in dem zu entwickelnden System in welchem Umfang behandelt werden.

Aufbauend auf den zu automatisierenden Geschäftsprozessen sind die Akteure des Systems abzugrenzen. Auch hier wurden bereits im Rahmen der Geschäftsprozessmodellierung die Akteure in den so genannten *Swimlanes* den einzelnen Aktivitäten zugeordnet. Durch die Festlegung der tatsächlichen Systemgrenzen muss aber auch die Menge der Akteure überarbeitet werden, da durch den Wegfall von Geschäftsabläufen diese Menge eingeschränkt werden kann.

Da wir neben Personen auch externe Hardware und Softwaresysteme (siehe [JBR99, Bre01, Kru00, Rat00]) mit dem zu erstellenden System interagieren, sind im Rahmen der Anwendungsfallmodellierung die Akteure um derartige externe Systeme und Hardware zu erweitern. Da im Rahmen der Geschäftsprozessmodellierung basierend auf den fehlenden Systemgrenzen die externen Systeme und Hardware noch nicht modelliert werden konnten, sind diese bei der Anwendungsfallmodellierung zu ermitteln und als Akteure aufzunehmen.

Für eine Bestimmung der Anwendungsfälle sind vorab diejenigen Geschäftsprozesse, die mit dem zu erstellenden System automatisiert werden sollen, zu überarbeiten. Da im Rahmen der Geschäftsprozessmodellierung auch Aktivitäten modelliert wurden, die nicht im System umgesetzt werden sollen, müssen in den Geschäftsprozessen die manuellen Aktivitäten eliminiert werden. In trivialen Fällen können die manuellen Aktivitäten aus den Abläufen entfernt werden. Andernfalls sind die Prozesse abzuändern und gegebenenfalls zu erweitern.

Die einzelnen Aktivitäten der Geschäftsprozesse können nun den interagierenden Akteuren zugeordnet werden. In diesem Schritt werden die Aktivitäten der Geschäftsprozesse in Anwendungsfälle verfeinert und in Anwendungsfalldiagrammen wird der Zusammenhang zwischen den Anwendungsfällen dargestellt. Falls die Aktivitäten keinen zusammenhängenden und vollständigen Ablauf darstellen, müssen die Aktivitäten weiter unterteilt oder zusammengefügt werden, sodass sie in Anwendungsfälle transformiert werden können.

Durch die Definition der Systemgrenzen können aus dem Domänenmodell diejenigen Objekte entfernt werden, die außerhalb der Systemgrenzen liegen und mit dem System nicht in Verbindung stehen. Dabei ist zu beachten, dass für eine Kommunikation mit externen Systemen Schnittstellen geschaffen werden müssen und dass hierzu Klassen zum Domänenmodell hinzuzufügen sind. Weiterhin sind bei der Verfeinerung des Domänenmodells Gemeinsamkeiten bei den Klassen zu erarbeiten und die Assoziationen anzupassen.

Die aus den Aktivitäten der Geschäftsprozesse ermittelten Anwendungsfälle sind auszuarbeiten und zu beschreiben (siehe hierzu Abschnitt 7.1.3). Die Daten, die im Anwendungsfall benötigt, erstellt und/oder manipuliert werden, sind zu erfassen und die geeigneten Datentypen im Domänenmodell zu ergänzen. Der Ablauf mit seinen Varianten wird textuell beschrieben. Für eine iterative Entwicklung können die Anwendungsfälle priorisiert werden, d.h., es wird festgelegt, in welchen Ausbaustufen die Anwendungsfälle umgesetzt werden.

Anwendungsfall Korrekturbuchung

Akteure
Team Worker

Input
AccountingID, Modified Accounting Data

Output
Accounting Data, Acknowledgement

Modifizierte Klassen des Anwendungskerns
Accounting

Beschreibung
Das System erhält vom Akteur *Team Worker* einen Buchungsidentifier (z.B. von einer vorausgehenden Suche). Die interne Verwaltungseinheit des Systems für Buchungen (*Accounting Manager*) sucht die entsprechende Buchung im System und zeigt sie dem Akteur *Team Worker* an. Der *Team Worker* modifiziert die Daten und das System speichert die Aktualisierungen. Am Ende gibt das System eine Bestätigung an den Akteur *Team Worker* zurück.

Varianten

1. Das System kann die Buchung zum vorgegebenen Buchungsidentifier nicht finden. Das System gibt eine negative Bestätigung an den Akteur *Team Worker* zurück.
2. Dem Akteur *Team Worker* werden die Daten der gesuchten Buchung angezeigt, er verändert jedoch keine Daten. Das System muss in diesem Fall keine Daten speichern. Es wird eine positive Bestätigung an den Akteur zurückgegeben.
3. Der Akteur *Team Worker* gibt ungültige Werte (falscher Datentyp, Überschreitung der Minimum- oder Maximumgrenzen, falsche Datumsangaben, etc.) ein. Das System informiert den Benutzer darüber und wartet auf neue Eingaben.

Abbildung 7.1: Beispiel eines Anwendungsfalls: Korrekturbuchung (adjustment posting)

7.1.3 Produktartefakte der Anwendungsfallmodellierung

Die zentralen Produktartefakte der Anwendungsfallmodellierung sind

- das *Anwendungsfallmodell*, in dem die Systemanforderungen strukturell und grafisch beschrieben werden und
- das *erweiterte Domänenmodell*, welches die statischen Konzepte beschreibt.

Im *Anwendungsfallmodell* werden die Anforderungen an das System in folgender Weise beschrieben:

- Die funktionalen Anforderungen werden aus den einzelnen Aktivitäten der Ablaufdiagramme der Geschäftsprozessmodellierung abgeleitet und in *Anwendungsfällen* strukturell beschrieben. Zu jedem Anwendungsfall ist der Name des Anwendungsfalls, die Ein- und Ausgabedaten, die zu modifizierten Klassen des statischen Anwendungskerns (Domänenmodell), eine Ablaufbeschreibung sowie eine Beschreibung der Ablaufvarianten textuell zu spezifizieren.

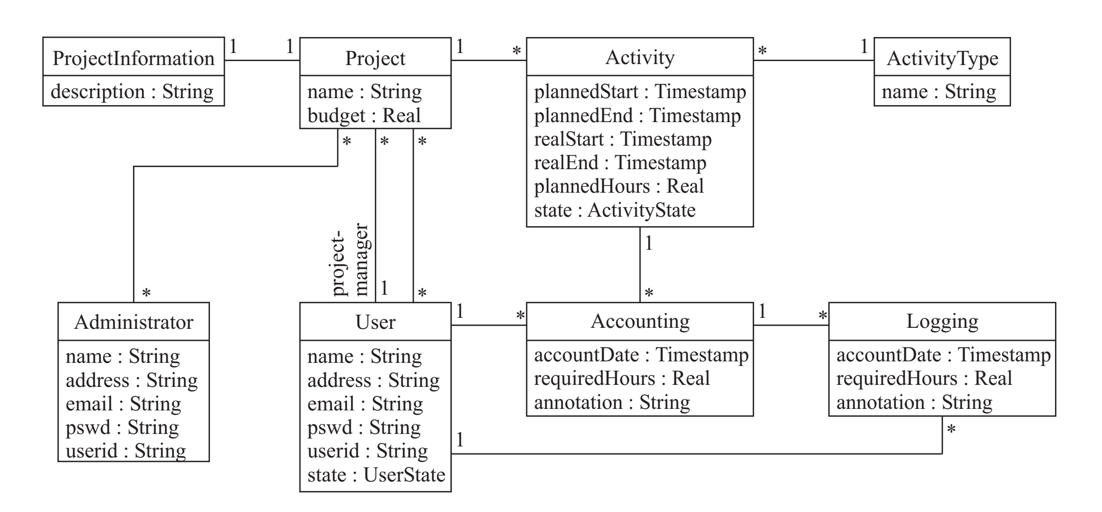


Abbildung 7.2: Domänenmodell der Anwendungsfallmodellierung

- *Anwendungsfalldiagramme* geben den Zusammenhang zwischen Anwendungsfällen und den involvierten Akteuren sowie die Zusammenhänge zwischen Anwendungsfällen untereinander wieder.
- In der *Anforderungsergänzung* werden nach [Kru00, Rat00] Anforderungen beschrieben, die über die Anwendungsfälle hinaus gehen. Darunter fallen gesetzliche Anforderungen und Regelungen, Qualitätsattribute (wie zum Beispiel Anwendbarkeit, Verfügbarkeit, Performance, Unterstützbarkeit), Umgebungsanforderungen (Betriebssysteme, Kompatibilität) oder Designanforderungen.
- Im *UI-Prototyp* wird ein Prototyp der Benutzerschnittstelle erstellt.

Das *erweiterte Domänenmodell* ist eine verfeinerte Version des Domänenmodells aus der Geschäftsprozessmodellierung. Das Domänenmodell ist an die festgelegten Systemgrenzen anzupassen. In den Klassendiagrammen sind Gemeinsamkeiten in den Klassen zu identifizieren und die Assoziationen anzupassen. Im Rahmen des Entwurfs des Anwendungskerns sind auch die fachlichen Datentypen, wie Währungen, Datumsangaben, etc. zu identifizieren.

Abbildung 7.1 zeigt die strukturelle Beschreibung des Anwendungsfalls *Korrekturbuchung* aus der *TimeTool*-Fallstudie. Ein Anwendungsfalldiagramm der Fallstudie ist bei der Einführung der Fallstudie *TimeTool* in Kapitel 3 dargestellte (siehe Abbildung 3.2).

Das verfeinerte Klassendiagramm des Domänenmodells aus der Geschäftsprozessmodellierung ist in Abbildung 7.2 zu sehen. Projektmitarbeiter und Projektleiter können in eine Klasse zusammengefasst werden. Gemeinsam mit der in Abschnitt 6.4.2 eingeführten Klasse *Logging* ergibt sich das in Abbildung 7.2 dargestellte Klassendiagramm.

7.2 Modellierung der Authentifikation

Bisher haben wir im Rahmen der Autorisierung den berechtigten Zugriff auf Daten und Aktivitäten betrachtet. Im Folgenden gehen wir auf die eng mit der Autorisierung verbundene Authentifikation ein, bei der die ausführende Entität eines Zugriffs anhand ihrer charakteristischen Eigenschaften oder anhand einer eindeutigen Identität vorab überprüft werden muss.

Nach der Authentifikation stehen der ausführenden Entität, d.h., einem Benutzer oder einem weiteren System, eine Menge von Aktivitäten zur Verfügung, für die sie autorisiert ist. Hierzu untersuchen wir im Folgenden, wie in den verfeinerten Abläufen der Geschäftsprozessmodellierung zugriffssicherer Systeme die Authentifikation in Verbindung mit so genannten Sitzungen (engl. *sessions*) modelliert werden und wie wir Sitzungen in den bereits modellierten Abläufen beschreiben können. Aus diesen Sitzungen leiten wir so genannte Authentifikationsanwendungsfälle ab und integrieren diese in die Abläufe.

7.2.1 Authentifikation

Bei der Authentifikation überprüfen wir die Echtheit und Glaubwürdigkeit einer ausführenden Entität anhand ihrer eindeutigen Identität und ihrer charakteristischen Eigenschaften (vgl. [Eck03]). Ausführende Entitäten sind im Rahmen der von uns betrachteten Systeme Akteure als Instanz von Rollen sowie weitere externe Systeme, die mit dem zu entwickelnden System interagieren.

In herkömmlichen Systemen basiert eine Identifikation auf der Vergabe von eindeutigen Benutzerkennungen oder Benutzernamen. Charakteristische Eigenschaften zum Nachweis der Identität sind beispielsweise Passworte, deren Kenntnis der Benutzer beim Systemzugang nachweisen muss, oder biometrische Merkmale wie Fingerabdrücke. Im Rahmen der Betrachtung von offenen Systemen wird diese einseitige Authentifikation (siehe auch [RE99], einseitige Authentisierung) zunehmend durch eine gegenseitige Authentifikation abgelöst, bei der die Glaubwürdigkeit beider an der Kommunikation beteiligter Partner vorab überprüft wird. Diese gegenseitige Authentifikation wird vor allem dann gefordert, wenn offene Systeme über ein unsicheres Transportmedium kommunizieren, wie beispielsweise über das Internet oder über eine Funkverbindung, wie etwa Wireless-LAN. Zur Überprüfung der gegenseitigen Authentifikation werden kryptografische Protokolle eingesetzt.

7.2.2 Authentifikation und Sitzungen

Die Authentifikation ist eine Voraussetzung, damit Akteure oder externe Systeme sicherheitskritische Daten lesen oder schreiben sowie sicherheitskritische Aktivitäten ausführen können. Sie stellt somit eine gesonderte Aktivität dar, die vor der Ausführung von derartigen Zugriffen erfolgreich ausgeführt werden muss. Es kann jedoch eine Reihe von Aktivitäten und Datenzugriffen geben, deren Ausführung nicht beschränkt ist. Beispielsweise können bei Web-basierten Bibliothekssystem anonyme Benutzer Such- und Informationsdienste nutzen. Vor der Bestellung oder Reservierung eines Buches müssen sie sich jedoch im System authentifizieren.

Innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme haben wir die Abläufe bereits exemplarisch modelliert. Im Rahmen der allgemeinen Anwendungsfallmodellierung wurden die exemplarischen Abläufe derart verfeinert, dass manuell zu verrichtende Aktivitäten aus den Abläufen entfernt wurden. In den exemplarischen Abläufen sind nun innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme die Sitzungen der verschiedenen, am Ablauf beteiligten Akteure zu ermitteln. Eine Sitzung stellt hier eine exemplarische Folge von Aktivitäten dar, die ein Akteur oder ein externes System ausführt. Für eine Teilmenge der Aktivitäten kann eine Überprüfung der Autorisierung der Akteure notwendig sein, sodass in

diesen Fällen vorab eine Authentifikation durchzuführen ist. Bei Sitzungen unterscheiden wir bezüglich der Authentifikation folgende Möglichkeiten:

- Die Aktivitäten der Sitzung dürfen alle nur nach einer vorangestellten, erfolgreichen Authentifikation stattfinden. Der Akteur oder das externe System bleiben über die gesamte Sitzung authentifiziert.
- Zu Beginn einer Sitzung ist keine Authentifikation notwendig, da vorab anonyme Aktionen durchgeführt werden, wie zum Beispiel das Suchen eines Buches innerhalb eines Bibliotheksystems. Für weitere Bearbeitungsschritte wird jedoch eine Authentifikation erforderlich. So muss sich beispielsweise ein Bibliotheksbenutzer vor der Bestellung oder Ausleihe eines Buches authentifizieren.
- Eine Authentifizierung hat bereits zu Beginn oder innerhalb der Sitzung stattgefunden. Für spezielle sicherheitskritische Aufgaben wird jedoch eine zusätzliche Authentifikation notwendig. Beispielsweise können Löschvorgänge in einem Bibliotheksverwaltungssystem nur dann ausgeführt werden, wenn sich der Administrator für diesen Vorgang speziell authentifiziert hat.
- Innerhalb der Sitzung ist keine Authentifikation notwendig. Die Sitzung stellt nur eine zusammengehörende Folge von Aktivitäten dar, die von einem anonymen Akteur ausgeführt wird.

Im Rahmen unserer Sicherheitsanalyse sind für uns die ersten drei Sitzungstypen von Interesse. Für diese Typen stellen wir im Folgenden geeignete Beschreibungstechniken vor.

7.2.3 Beschreibungstechniken zum Schutzziel Authentifikation

Um diese Arten von Authentifikationen und Sitzungen innerhalb der Prozessabläufe modellieren zu können, führen wir für Aktivitäten weitere Stereotypen ein.

Zur Unterscheidung von einseitiger und gegenseitiger Authentifizierung führen wir für das Schutzziel Authentifikation zwei Stereotypen ≪*Authenticity*≫ und ≪*MutualAuthenticity*≫ ein, wie sie in Abbildung 7.3 dargestellt sind. Für Aktivitäten, die mit dem Stereotyp ≪*Authenticity*≫ gekennzeichnet sind, wird gefordert, dass der ausführende Akteur oder ein externes System vorab am System authentifiziert sein muss. Aktivitäten, die mit dem Stereotyp ≪*MutualAuthenticity*≫ gekennzeichnet sind, erfordern, dass der ausführende Akteur oder ein externes System sowie das modellierte System vorab gegenseitig ihre Identitäten überprüfen.

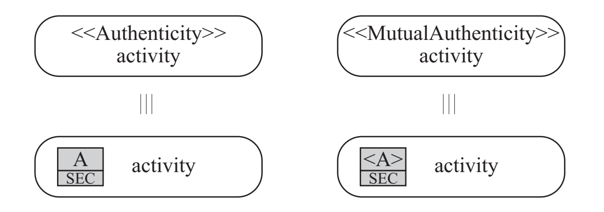


Abbildung 7.3: Stereotypen zur einfachen und beidseitigen Authentifikation

Bei der Untersuchung der Abläufe wird nun jede Aktivität bezüglich des Schutzziels Authentifikation untersucht. Falls der ausführende Akteur, der durch die Swimlane zugeordnet wurde, zur Ausführung der Aktivität einseitig oder gegenseitig am System authentifiziert werden muss, wird die Aktivität mit dem entsprechenden Stereotyp gekennzeichnet.

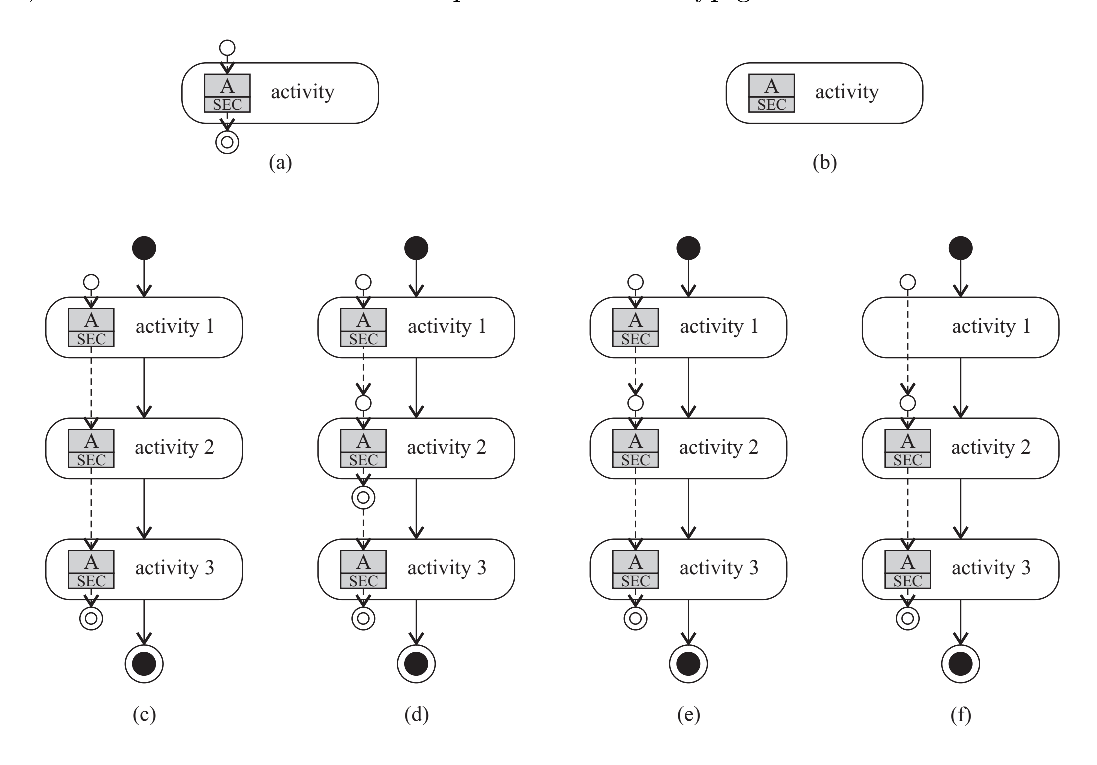


Abbildung 7.4: Authentifikation und Sitzungen in Abläufen

Für aufeinander folgende Aktivitäten ist es zumeist nicht erforderlich und nicht geeignet, dass der Akteur oder ein externes System für jede Aktivität separat authentifiziert werden muss. Aus diesem Grund haben wir im Abschnitt 7.2.2 bereits Sitzungen und verschiedene Fälle eingeführt, in denen ein Akteur nur zu unterschiedlichen Aktivitäten einer Sitzung authentifiziert werden muss. Um derartige Sitzungen und Authentifikationen innerhalb der Ablaufdiagramme exemplarisch darstellen zu können, führen wir zwischen den Schutzziel-Icons zur Authentifikation besondere Assoziationen mit Start- und Endsymbolen ein.

Wird die Authentifikation nur für eine einzelne Aktivität innerhalb eines Ablaufs gefordert, findet die Authentifikation unmittelbar vor der Ausführung der Aktivität statt und wird anschließend sofort beendigt. Abbildung 7.4 (a) zeigt hierzu die speziellen Symbole zur Darstellung des Beginns und Endes einer Authentifikation. Analog zu einem Aktivitätsfluss innerhalb eines Aktivitätsdiagramms führen wir hier einen eigenen Fluss für die Authentifikation ein.

In diesem trivialen Fall der Authentifikation, indem die Sitzung nur aus einer einzigen Aktivität besteht und der Beginn und das Ende der Authentifikation aus dem Authentifikations-Icon erkennbar sind, können wir in diesem Spezialfall die Kennzeichnung des Beginns und Endes auch weglassen (siehe Abbildung 7.4 (b)). Die alleinige Verwendung des Schutzziels *Authentifikation* ohne weitere Darstellung einer Sitzung fordert somit, dass für diese spezielle Aktivität eine eigene Authentifikation stattzufinden hat.

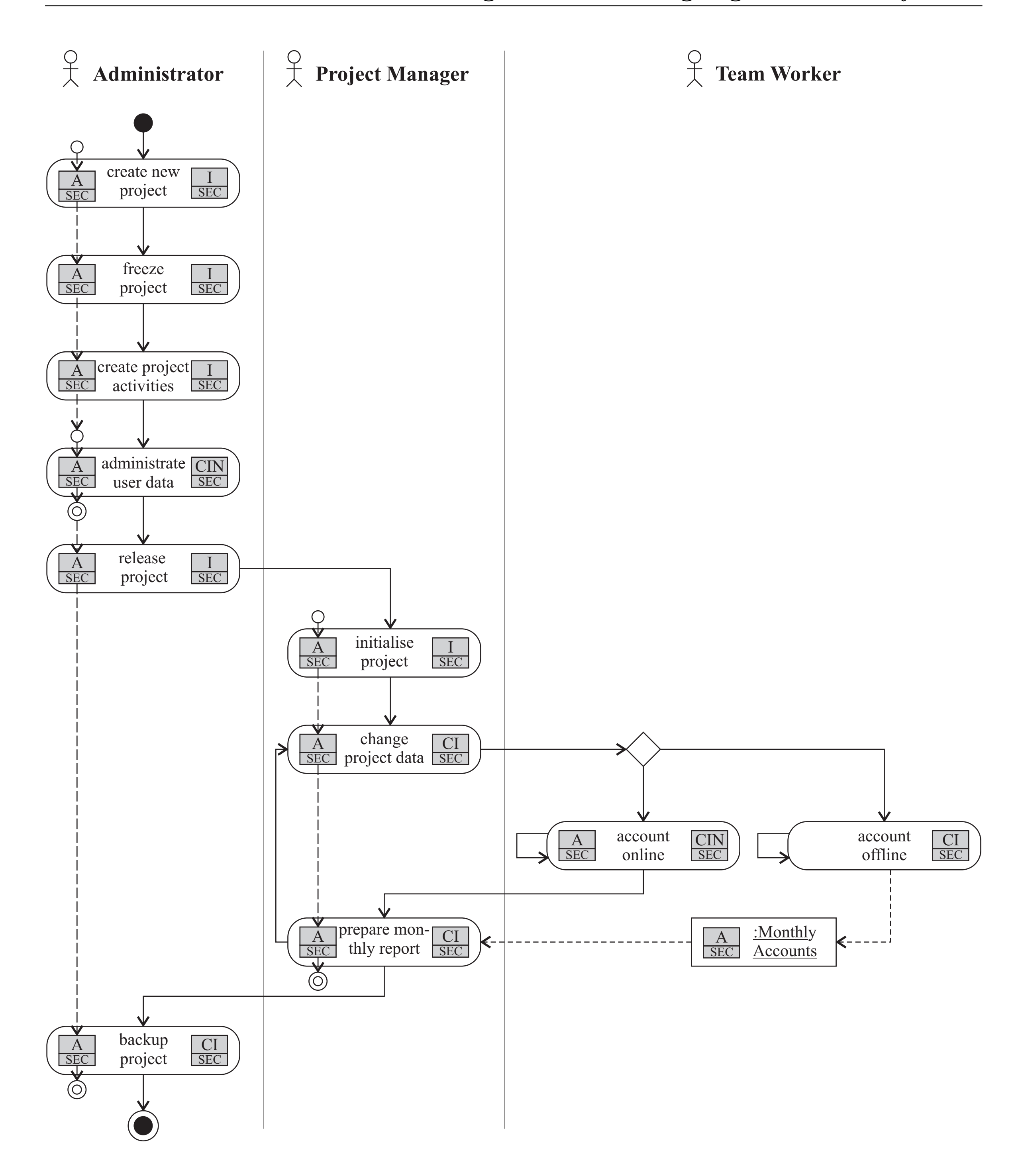


Abbildung 7.5: Ablauf des TimeTool-Systems mit Authentifikation und Sitzungen

Werden innerhalb einer Sitzung mehrere Aktivitäten ausgeführt, für die eine Authentifizierung notwendig ist, kennzeichnen wir zu Beginn der ersten Aktivität den Beginn der Authentifikation und nach der letzten Aktivität das Ende der Authentifikation (vgl. Abbildung 7.4 (c)). In diesem Fall hat vor der Ausführung der Sitzung mit ihren Aktivitäten eine Authentifikation stattzufinden.

Der bereits erwähnte Sonderfall, dass innerhalb einer Sitzung für spezielle Aufgaben eine zusätzliche gesonderte Authentifikation erforderlich wird, ist in Abbildung 7.4 (d) dargestellt. Für diejenigen Aktivitäten, die eine eigene Authentifikation benötigen, wird im Authentifikationsfluss zusätzlich Beginn und Ende einer Authentifikation gekennzeichnet. Die explizite Authentifikation kann auch mehrere Aktivitäten überdecken und hierarchisch geschachtelt werden. In derartigen Fällen setzt der Akteur nach Beendigung der speziellen Authentifikation die Ausführung der Aktivitäten mit seiner ursprünglichen Authentifikation fort, sodass beispielsweise beim wiederholten Aufruf der Aktivität mit eigener Authentifikation diese erneut durchzuführen ist. Im Gegensatz dazu zeigt Abbildung 7.4 (e) den Fall, dass Akteur sich innerhalb einer Sitzung für eine weitere Aktivität speziell authentifiziert, anschließend diese spezielle Authentifikation aber bis zum Ende der Session weiter ausführt. Mit Beendigung der Session werden beide Authentifikationen beendet.

Abbildung 7.4 (f) zeigt einen weiteren Sonderfall der Authentifikation, bei dem der Akteur innerhalb der Sitzung zunächst eine Aktivität ohne Authentifikation ausführt, dann jedoch im weiteren Verlauf der Sitzung eine Authentifikation durchführt.

In Abbildung 7.4 wurde in allen Beispielen stets die einseitige Authentifizierung verwendet. In allen Darstellungen kann entsprechend die beidseitige Authentifizierung verwendet werden. Für den Fall, dass innerhalb einer einseitigen Authentifikation eine beidseitige Authentifikation gefordert wird oder umgekehrt, muss der Ablauf aus Abbildung 7.4 (d) bzw. 7.4 (e) angepasst werden.

Abbildung 7.5 zeigt einen Ablauf der *TimeTool*-Fallstudie, bei dem im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme die Akteure verfeinert sowie Sitzungen und Authentifikationsschutzziele ergänzt wurden. Im Ablauf wurden drei Sitzungen identifiziert. Für den *Team Worker* wurde eine Sitzung mit nur einer einzigen Aktivität ermittelt. Die Sitzungen von *Project Manager* und *Administrator* wurden mittels der eingeführten Symbole annotiert. Alle drei Sitzungen erfordern, dass vor Ausführung der ersten Aktivität der zugehörige Akteur authentifiziert wird und nach Abschluss der letzten Aktivität innerhalb der Sitzung vom System abgemeldet wird. Vor Ausführung der Aktivität *administrate user data* muss sich der *Administrator* explizit nochmals authentifizieren, nach der Ausführung wechselt er wieder zu seiner ursprünglichen Identität im System zurück.

7.2.4 Verfeinerung der Authentifikation und der Sitzungen

Aus den Aktivitäten, die wir mit dem Schutzziel Authentifikation annotiert haben, und der Modellierung der Sitzungen können wir die Abläufe derart verfeinern, dass wir eigene Aktivitäten zur Authentifikation in die Abläufe integrieren.

Hierzu fügen wir innerhalb der exemplarischen Ablaufdiagramme vor der ersten Aktivität, für die der Akteur oder ein externes System authentifiziert sein müssen, eine Aktivität *login* ein, am Ende der Sitzung eine Aktivität *logout*. Sollte innerhalb der Sitzung einer der oben erläuterten Sonderfälle auftreten, so ist gegebenenfalls für spezielle Aktivitäten eine zusätzliche Authentifikation einzufügen.

In Abbildung 7.6 (a), (b) wurden die Abläufe aus Abbildung 7.4 (c), (d) verfeinert. Im Abbildung 7.6 (b) ist für die Aktivität *activity5* eine spezielle Authentifikation erforderlich, sodass hierfür die beiden Aktivitäten $login_S$ und $logout_S$ eingefügt wurden.

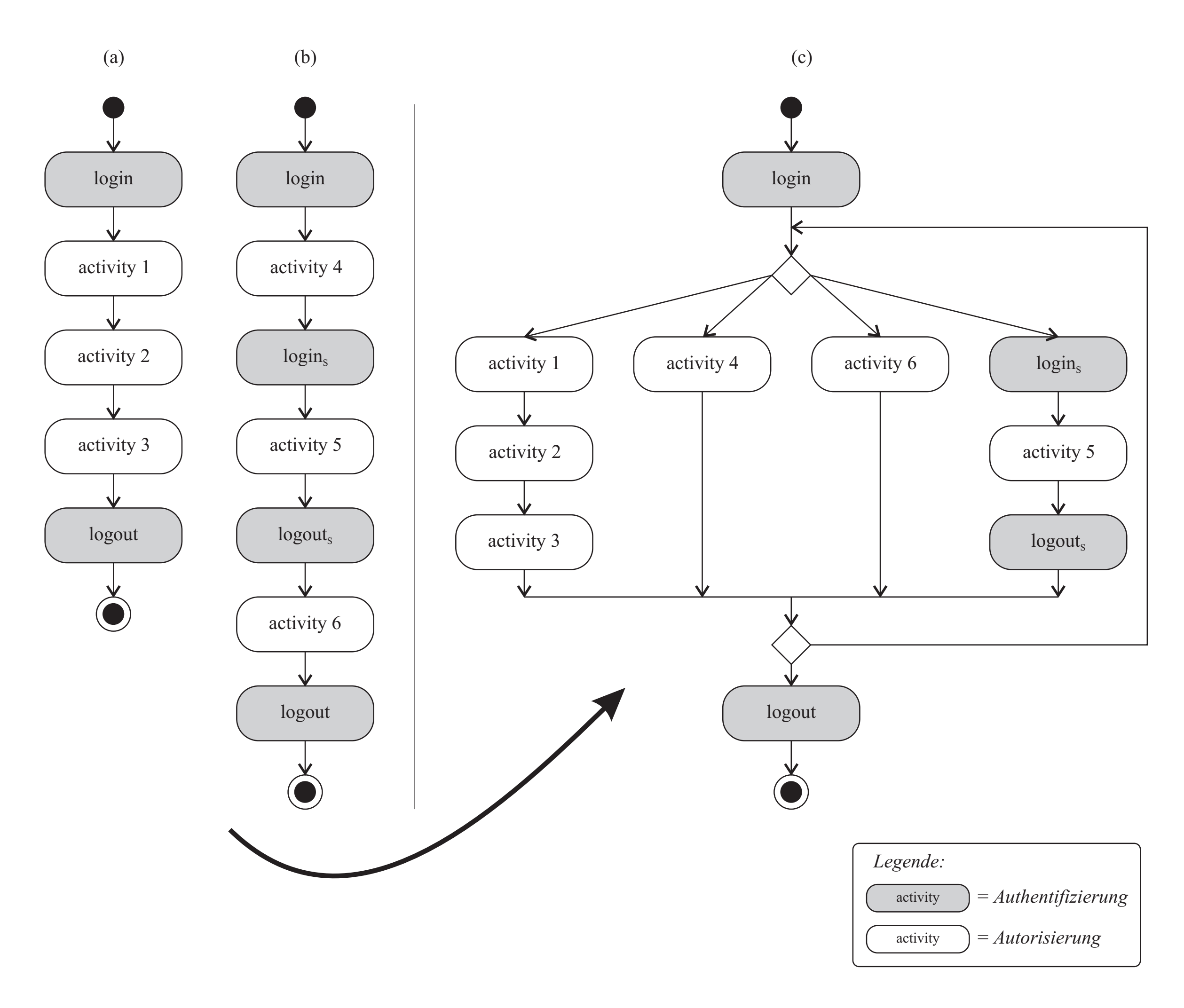


Abbildung 7.6: Verfeinerung der Authentifikation und Sitzungen

Zur Bestimmung der notwendigen Authentifikationsanwendungsfälle ist weiterhin zu untersuchen, inwiefern aus den verschiedenen exemplarischen Abläufen die Aktivitäten zur Authentifikation durch einheitliche *Login*-Anwendungsfälle realisiert werden können. Für die eingeführten Spezialfälle kann gegebenenfalls die gewöhnliche Authentifizierung nicht wieder verwendet werden, da sich das System bereits ein Akteur angemeldet hat und dieser Kontext nach Beendigung der speziellen Authentifikation wiederherzustellen ist.

Zur Ermittlung der gemeinsamen und speziellen Authentifikationen fügen wir die eben verfeinerten Ablaufdiagramme nach folgenden Heuristiken zusammen:

- Werden innerhalb von durchgehend authentifizierten Abläufen, d.h. in Abläufen, bei denen ein Akteur oder ein externes System zu Beginn authentifiziert und am Ende des Ablaufs abgemeldet wird, einheitlich nur einseitige oder gegenseitige Authentifikation verwendet, so können die einzelnen Abläufe so zusammengefügt werden, dass zu Beginn eine gemeinsame Authentifizierung für alle Akteure und externe Systeme stattfindet (vgl. hierzu allgemeine Login- und Logout-Aktivitäten am Beginn und Ende der Abläufe in Abbildung 7.6).

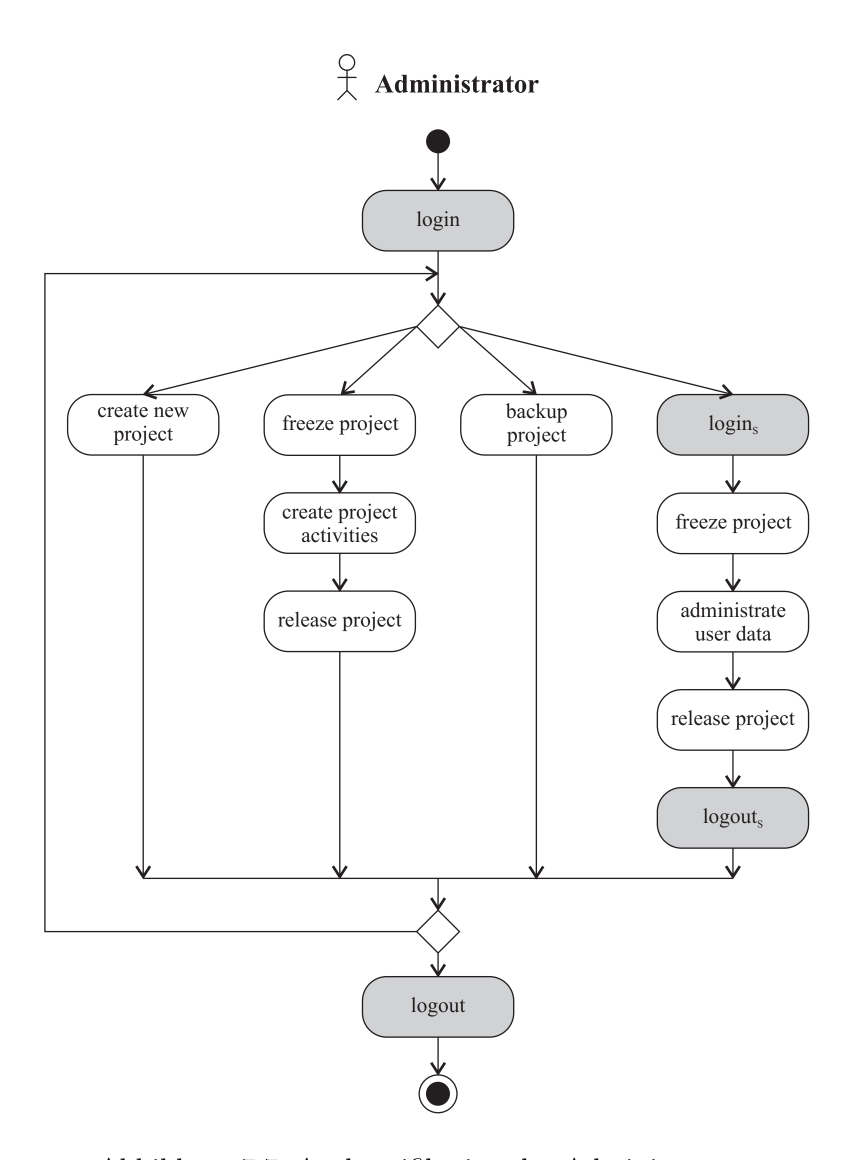


Abbildung 7.7: Authentifikation des Administrators

- In den exemplarischen Ablaufdiagrammen werden die Aktivitäten nach der Authentifizierung sequenziell ausgeführt. Einige Aktivitäten oder Teilsequenzen dieser Sequenzen können auch eigenständig, d.h. losgelöst aus der Sequenz, ausgeführt werden, sodass vorab ebenfalls eine eigene Authentifizierung notwendig wird. Diejenigen Aktivitäten oder Teilsequenzen können dann nach einem Login alternativ ausgeführt werden. Beispiele sind hierzu beispielsweise die Aktivitäten *activity4* und *activity6* in Abbildung 7.6.

- Abläufe, die eine zusätzliche, spezielle Authentifikation erfordern, müssen als zusätzlicher Pfad nach der allgemeinen Authentifikation zu den bisherigen Abläufen hinzugefügt werden. In Abbildung 7.6 wurde hierzu die Teilsequenz bestehend aus $login_S$, *activity5* und $logout_S$ hinzugefügt. Vor und nach den Aktivitäten zur zusätzlichen, speziellen Authentifikation können innerhalb des Pfades auch weitere Aktivitäten notwendig sein, d.h., es ist nicht zwingend erforderlich, dass die spezielle Authentifizierung unmittelbar

nach der allgemeinen Authentifizierung stattfindet und die Abmeldung im speziellen Fall unmittelbar vor dem allgemeinen Logout.

- Kommen im System sowohl Abläufe mit einseitiger sowie beidseitiger Authentifikation vor, so können diese nicht dieselbe *login*-Aktivität verwenden. Hierzu sind im kombinierten Ablaufdiagramm zwei alternative Aktivitäten für die entsprechenden Authentifikationen einzuführen.

Das durch diese Methode entstehende Ablaufdiagramm erhebt nicht den Anspruch, dass es alle Abläufe vollständig modelliert. Ziel dieses Ablaufdiagramms liegt darin, die notwendigen Aktivitäten zur Authentifikation zu ermitteln und Gemeinsamkeiten festzustellen. Wir können nun zwischen denjenigen Aktivitäten unterscheiden, die zur Überprüfung der Authentifikation eingeführt wurden und denjenigen, die vorab mit dem Stereotyp *Authenticity* annotiert wurden. Bei allen Aktivitäten, die mit dem Schutzziel *Authenticity* gekennzeichnet wurden, muss vor der Ausführung überprüft werden, inwiefern der ausführende Akteur autorisiert ist, die Authentifikation wird jeweils separat ausgeführt und im Folgenden auch separat spezifiziert.

Aus den Aktivitäten zur Authentifikation sind im weiteren Verlauf der Anwendungsfallmodellierung zugriffssicherer Systeme die Anwendungsfälle zur Authentifikation (Anwendungsfälle zum Login und Logout) auszuarbeiten (siehe Abschnitt 7.3).

Abbildung 7.7 zeigt die Verfeinerung des in Abbildung 7.5 dargestellten Ablaufs zur *Time-Tool*-Fallstudie. Wir beschränken und dabei auf den Akteur *Administrator*. Die im exemplarischen Ablauf sequenziell ausgeführten Aktivitäten *create new project*, *create project activities*, *backup project* und *administrate user data* können bei der Anwendung in beliebiger Reihenfolge ausgeführt werden. Alle diese Aktivitäten können nur nach einer erfolgreichen Authentifizierung am System aufgerufen werden, bei der Aktivität *administrate user data* muss vorab nochmals eine spezielle Authentifikation ausgeführt werden. Die grau hinterlegten Aktivitäten stellen die Aktivitäten zur Authentifizierung, bei den weiß hinterlegten Aktivitäten muss jeweils nur die Autorisierung überprüft werden.

7.3 Entwicklung der Sicherheitsanwendungsfälle

Wie wir bereits in Abschnitt 7.1 beschrieben haben, sind im Rahmen der Anwendungsfallmodellierung Aktivitäten der Geschäftsprozesse in Anwendungsfälle überzuführen. Es sind nur diejenigen Aktivitäten zu transformieren, die durch das zu entwickelnde System realisiert werden sollen. Alle zu realisierenden Anwendungsfälle beschreiben die Gesamtfunktionalität des zu entwickelnden Systems.

Bei der Geschäftsprozessmodellierung zugriffssicherer Systeme haben wir den sicherheitskritischen Aktivitäten bereits Schutzziele zugeordnet, Bedrohungen und Risiken ermittelt sowie Maßnahmen zur Abwehr entworfen. Im Rahmen der Entwicklung der Sicherheitsanwendungsfälle sind nun die Schutzziele aus den Aktivitäten der Geschäftsprozesse in die Anwendungsfälle zu integrieren und die ermittelten Bedrohungen, Risiken und Maßnahmen anzupassen. Da die Betrachtung der Zugriffssicherheit auch weitere funktionale Anforderungen mit sich bringt, muss die Beschreibung der funktionalen Anforderungen des Systems um so genannte Sicherheitsanwendungsfälle ergänzt werden.

Auf die Erweiterung von Anwendungsfällen um Sicherheitsaspekte gehen wir in Abschnitt 7.3.1 ein. Die Ergänzung von Sicherheitsanwendungsfällen wird in Abschnitt 7.3.1 eingeführt.

7.3.1 Erweiterung der Anwendungsfälle um Aspekte der Zugriffssicherheit

Aktivitäten der Geschäftsprozesse werden auf Anwendungsfälle abgebildet und darin weiter verfeinert. Hierbei kann eine *n:m*-Zuordnung auftreten, d.h., eine oder mehrere Aktivitäten können auf einen oder mehrere Anwendungsfälle abgebildet werden.

Wurden in einer Aktivität Schutzziele spezifiziert, so müssen ein oder mehrere um Sicherheitsaspekte erweiterte Anwendungsfälle entwickelt werden. Dabei sind die Schutzziele auf die erweiterten Anwendungsfälle zu transformieren. Im Fall einer *1:1*-Zuordnung können die Schutzziele direkt auf den Anwendungsfall übernommen werden. Wird eine Aktivität auf mehrere Anwendungsfälle transformiert (*1:n*-Zuordnung), so sind die Schutzziele auf die Anwendungsfälle zu übertragen und es ist zu überprüfen, inwiefern die spezifizierten Schutzziele für jeden einzelnen Anwendungsfall zutreffen. Bei der Zuordnung von mehreren Aktivitäten zu einem oder mehreren Anwendungsfällen können in den Aktivitäten unterschiedliche Schutzziele definiert worden sein. In diesen Fällen (*m:1*- oder *m:n*-Zuordnung) muss für alle n Anwendungsfälle erneut überprüft werden, inwiefern alle Schutzziele aus den m Aktivitäten für die einzelnen Anwendungsfälle Bedrohungen hervorrufen.

In einem Anwendungsfall sind die Ein- und Ausgabedaten sowie die modifizierten Daten beschrieben. Bei der Ermittlung der Schutzziele des Anwendungsfalls sind neben den Schutzziele aus den Aktivitäten auch die Schutzziele, die auf diesen Daten spezifiziert sind, einzubeziehen. Ergeben sich aus den Daten weitere Anforderungen an die Zugriffssicherheit, so sind die Schutzziele zur erweiterten Beschreibung des Anwendungsfalls hinzuzunehmen.

Wurde für eine Aktivität das Schutzziel *Authentifikation* ermittelt und wurde im Abschnitt 7.2 bei der Modellierung der Authentifikation festgelegt, dass die Autorisierung zu überprüfen ist, so ist im Anwendungsfall hierfür eine Vorbedingung hinzuzufügen. Vorbedingungen in Anwendungsfällen wurden bereits in [FH97b] diskutiert und als Element einer erweiterten Anwendungsfallbeschreibung eingeführt.

Beschreibung der Anforderungen der Zugriffssicherheit

Für die Beschreibungen von Anforderungen an die Zugriffssicherheit wird die strukturelle Anwendungsfallbeschreibung aus [Bre01] um einen Abschnitt *“Sicherheit”* ergänzt. Darin werden Schutzziele, Vorbedingungen, Modifikationen am Ablauf sowie weitere Änderungen am Anwendungsfall, die aus der Ermittlung der Zugriffssicherheit hervorgehen, beschrieben. Ergeben sich aus der Ermittlung der Zugriffssicherheit auch Varianten zum Ablauf, so sind diese der Beschreibung der Varianten hinzuzufügen.

In Abbildung 7.8 ist die Erweiterung der Anwendungsfallbeschreibung zum Anwendungsfall *Korrekturbuchung* (vgl. Abbildung 7.1) aus unserer *TimeTool*-Fallstudie dargestellt. Schutzziele aus der Modellierung der Geschäftsprozesse sowie aus der Modellierung der Authentifikation werden als Anforderungen *A1* und *A2* im Abschnitt zur Beschreibung der Sicherheitsanforderungen aufgenommen. Die Anforderung *A3* wurde bei der Erarbeitung des Anwendungsfalls erkannt und aufgenommen. Bei der Modellierung der Bedrohungen und Risiken in der Geschäftsprozessmodellierung zugriffssicherer Systeme in Abschnitt 6.4.2 wurde festgelegt,

Anwendungsfall Korrekturbuchung

(... Beschreibung aus Abbildung 7.1)

Sicherheit

A1 Das System muss die Eingabedaten vertraulich behandeln und die Integrität muss gewährleistet werden.

A2 Der Projektmitarbeiter kann den Anwendungsfall nur ausführen, wenn er im System authentifiziert ist. Vor der Ausführung des Anwendungsfalls Korrekturbuchung wird die Autorisierung überprüft.

A3 Der Web Browser und das *TimeTool*-System müssen sich Beginn der Transaktionen authentifizieren.

A4 Korrekturbuchungen werden vom System in einer Klasse Logging mitprotokolliert.

Varianten

4. Ist ein Projektmitarbeiter im System nicht authentifiziert, so kann er den Anwendungsfall Korrekturbuchung nicht ausführen. Schlägt die Autorisierung fehl, informiert das System den Benutzer über die fehlenden Benutzerrechte und bricht mit einer negativen Bestätigung ab.
5. Schlägt die Authentifikation des Web-Browsers am TimeTool-System fehl, wird eine Fehlermeldung zurückgegeben und der Anwendungsfall bricht mit einer negativen Bestätigung ab.

Abbildung 7.8: Sektion zur Sicherheit des Anwendungsfalls Korrekturbuchung

dass Änderungen an Buchungen zur Beweissicherung mitprotokolliert werden müssen (Bedrohung B013). Da im Anwendungsfall zur Korrekturbuchung Daten des Objekts *Accounting* verändert werden, wird die Anforderung zum Logging *A4* hinzugefügt.

Anpassung der Benutzerrechte

Im Benutzerrechtemodell sind bei der Geschäftsprozessmodellierung zugriffssicherer Systeme die Ausführungsrechte auf Aktivitäten festgelegt worden. Da jedoch die Aktivitäten in Anwendungsfälle verfeinert wurden, sind die Ausführungsrechte nun an die Anwendungsfälle anzupassen.

Da wir die Benutzerrechte in eine Benutzerrechtematrix ausgelagert haben, muss diese an die Anwendungsfälle angepasst werden. In Abhängigkeit von der Zuordnung von Aktivitäten zu Anwendungsfällen sind folgende Modifikationen der Benutzerrechtematrix notwendig:

(1:1) Wenn eine Aktivität genau einem Anwendungsfall zugeordnet wird, muss an der Benutzerrechtematrix keine Veränderung durchgeführt werden. Der Anwendungsfall übernimmt dabei den Bezeichner der Aktivität.

(1:n) Werden zu einer Aktivität mehrere Anwendungsfälle eingefügt, müssen Zeilen in die Benutzerrechtematrix aufgenommen werden. Die Berechtigungen können dabei von der Aktivität übernommen werden, jedoch ist eine Anpassung zu prüfen.

(m:1) Werden mehrere Aktivitäten durch einen einzigen Anwendungsfall verfeinert, so sind in der Benutzerrechtematrix Zeilen zu streichen. Die Berechtigung für den Anwendungsfall sind aus den Berechtigungen der Aktivitäten zu ermitteln.

(m:n) Werden mehrere Aktivitäten durch mehrere Anwendungsfälle verfeinert (jedoch nicht in einer 1:1-Zuordnung), so sind m Zeilen der ursprünglichen Benutzerrechtematrix durch n Zeilen zu ersetzen und anzupassen.

Falls die Benutzerrechte von Anwendungsfällen nicht *1:1* aus Aktivitäten übernommen werden können, so muss vor der Bestimmung der Schutzziele der Sicherheitsmikroprozess ausgeführt werden (siehe hierzu Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme in Abschnitt 7.5).

7.3.2 Ergänzung von Sicherheitsanwendungsfällen

Neben der Erweiterung von bestehenden Anwendungsfällen sind auch eigene Sicherheitsanwendungsfälle (siehe auch [Fir03]) zu modellieren, in welchen die Sicherheitsgrundfunktionen des Systems modelliert sind. In diesen Grundfunktionen wird die Basisfunktionalität der Sicherheitsmaßnahmen beschrieben. Diese Funktionen sind eine Art "Baukasten" von allgemeinen Maßnahmen zur Abwehr von Bedrohungen, müssen jedoch entsprechend den individuellen Anforderungen des zu konstruierenden Systems angepasst werden [Eck03]. Sicherheitsanwendungsfälle umfassen beispielsweise Funktionen zur Authentifikation, Autorisierung, Rechteverwaltung, Beweissicherung und Verschlüsselung. Die Funktionalität wird entsprechend der vorgestellten strukturellen Beschreibungstechnik erläutert. Detaillierte Protokolle und technische Einzelheiten sind im Rahmen der Analyse zugriffssicherer Systeme (vgl. Kapitel 8) und des Designs festzulegen. In Abbildung 7.9 sind die zusätzlichen Sicherheitsanwendungsfälle zu unserer *TimeTool*-Fallstudie dargestellt.

7.4 Evolution von Berechtigungen

In der Geschäftsprozessmodellierung zugriffssicherer Systeme haben wir bereits Zugriffsrechte auf die Objekte des Domänenmodells sowie Ausführungsrechte auf Aktivitäten der Geschäftsprozesse spezifiziert. Innerhalb der Sicherheitsanwendungsfälle wird das Domänenmodell verfeinert und die Aktivitäten in Anwendungsfällen spezifiziert. Auf die damit verbundene Verfeinerung der Benutzerrechte gehen wir in Abschnitt 7.4.1 ein.

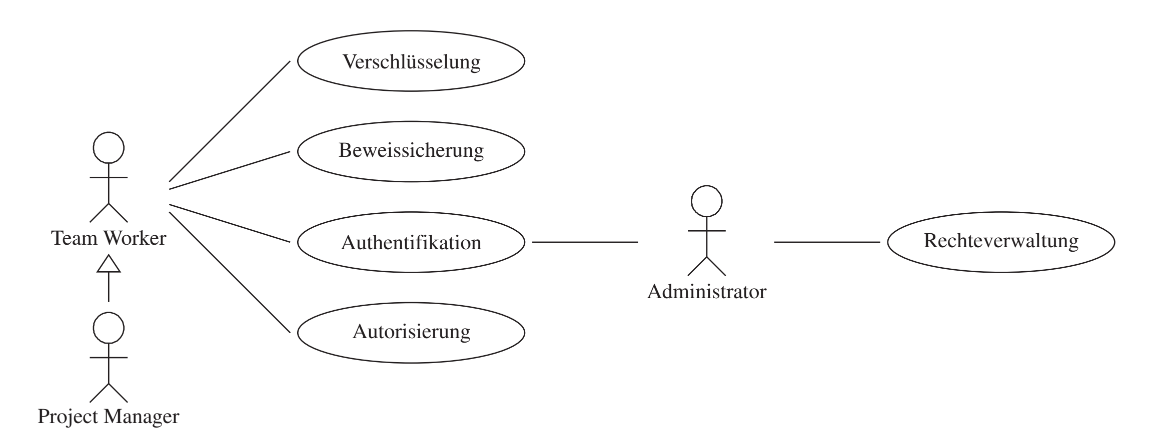


Abbildung 7.9: Sicherheitsanwendungsfälle der TimeTool-Fallstudie

Durch die Spezifikation der Abläufe in Anwendungsfällen wird es möglich, die Ausführungsrechte auf Aktivitäten aus den Zugriffsrechten zu berechnen. In Abschnitt 7.4.2 betrachten wir die Rahmenbedingungen und die Berechnung von Ausführungsrechten aus Zugriffsrechten.

7.4.1 Verfeinerung der Benutzerrechte

Die Benutzerrechte auf Klassen des Domänenmodells und Aktivitäten der Geschäftsprozesse sind im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme den neuen Strukturen und Ablaufeinheiten anzupassen, zu verfeinern und zu ergänzen.

Das Domänenmodell bei der Erarbeitung der Anwendungsfälle erweitert und optimiert. Hierzu sind die Berechtigungen in der Benutzerrechtematrix zu überarbeiten. Für neu hinzugefügte Klassen sind Berechtigungen mit aufzunehmen. Werden Klassen optimiert, beispielsweise durch die Einführung von Superklassen zur Modellierung der Gemeinsamkeiten, so sind die Berechtigungen für die Super- und Subklassen anzupassen. Auch bei der Änderung von Attributen der Klassen sind die Benutzerrechte zu überprüfen, da beispielsweise die Hinzunahme oder Wegnahme von sensitiven Daten zu einer Klasse die Benutzerrechte beeinflussen. Weiterhin kann eine Überarbeitung der Benutzerrechte auch durch die Änderung der Klassenstruktur hervorgerufen werden, so zum Beispiel durch die Auslagerung von Attributen in eigene Klassen und die damit verbundene Änderung von Assoziationen.

Die Benutzerrechte zur Ausführung von Aktivitäten und Abläufen sind innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme zu überarbeiten. Da zu Beginn der Anwendungsfallmodellierung zugriffssicherer Systeme festgelegt wird, welche Abläufe im zu entwickelnden System umgesetzt werden, sind nur diejenigen Benutzerrechte für die im System umzusetzenden Teile weiterhin zu modellieren. Die Benutzerrechte zu den nicht automatisierbaren Teilen sind aus der Berechtigungsmatrix zu streichen. Zu den Aktivitäten der Geschäftsprozesse sind die Anwendungsfälle festzulegen. Wie in Abschnitt 7.3 beschrieben, ist dabei eine *n:m*-Zuordnung möglich. In den Fällen einer *1:1*- oder *1:n*-Zuordnung können die Benutzerrechte einer Aktivität auf den einen Anwendungsfall oder die n Anwendungsfälle übernommen werden. Die Benutzerrechte müssen insbesondere bei der *m:1*-Zuordnung angepasst werden. Werden mehrere Aktivitäten mit verschiedenen Benutzerrechten auf einen einzelnen Anwendungsfall übertragen, so sind die Benutzerrechte zu diesem Anwendungsfall erneut zu spezifizieren.

Die Benutzerrechte sind auch an die Akteure anzupassen. Werden bei der Festlegung der Anwendungsfälle die Akteure des Systems verändert, d.h., neu geordnet oder Akteure werden hinzugefügt oder entfernt, so ist die Benutzerrechtematrix anzupassen. Für neue Akteure sind eigene Spalten einzufügen und die Benutzerrechte sind für die Domänenobjekte und Anwendungsfälle zu spezifizieren. Die Hinzunahme oder Wegnahme von Akteuren sowie die Änderung der Akteurhierarchie kann auch weitreichende Änderungen auf die bereits spezifizierten Benutzerrechte haben, da diese an die neue Menge und Hierarchie anzupassen sind.

Unabhängig von Änderungen an der Klassenstruktur, an Abläufen und an den Akteuren sind die Benutzerrechte im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme anzupassen. Aufgrund von der mittlerweile vorliegenden detaillierten Beschreibung von Systemanforderungen können die Benutzerrechte präziser beschrieben werden. So kann beispielsweise bei der Geschäftsprozessmodellierung zugriffssicherer Systeme festgelegt werden, dass ein Akteur A eine Aktivität x ausführen darf. Aufgrund von detailliertem Hintergrundwissen

kann das Benutzerrecht jedoch weiter eingeschränkt werden, so etwa auf einen bestimmten Zeitraum oder auf das Vorhandensein weiterer Objekte. Da wir zu den Benutzerrechten weitere Informationen hinzufügen, sprechen wir von einer *Verfeinerung* der Benutzerrechte.

7.4.2 Spezifikation und Überprüfung von Ausführungsrechten

Zu Beginn der Entwicklung zugriffssicherer Systeme werden die Benutzerrechte auf Objekten des Domänenmodells sowie Aktivitäten der Geschäftsprozesse getrennt voneinander spezifiziert (vgl. Kapitel 6.3). Da die Geschäftsprozesse jedoch auf den Objekten des Domänenmodells operieren, besteht zwischen den Spezifikationen der Benutzerrechte ein Zusammenhang. Insbesondere können durch diese getrennte Spezifikation der Benutzerrechte Inkonsistenzen entstehen.

Den Zusammenhang zwischen Zugriffsrechten, d.h., Benutzerrechten auf Objekten des Domänenmodells, und Ausführungsrechten, den Benutzerrechten auf Aktivitäten, können wir über die Ablaufbeschreibungen der Anwendungsfälle herstellen. Hinter Anwendungsfällen, welche einer oder mehrerer Aktivitäten eines Geschäftsprozesses zugeordnet sind, verbergen sich Abläufe. Diese Abläufe beschreiben letztendlich mit einer Menge von Methodenaufrufen und Kontrollstrukturen eine Menge aus Datenzugriffen auf Objekten des Domänenmodells und aus Aufrufen von Systemfunktionen, Berechnungsfunktionen oder Funktionen externer Systeme (wie Datenbankzugriffe, etc.). Will nun ein Akteur einen Anwendungsfall ausführen, so muss der Akteur für all diese Methodenaufrufe die notwendigen Benutzerrechte besitzen, andernfalls würde die Ausführung des Anwendungsfalls zwischendurch abbrechen. Fassen wir alle Berechtigungen, die ein Akteur bei der Ausführung eines Ablaufs benötigt, zusammen, so erhalten wir das Ausführungsrecht für einen Anwendungsfall. Wir können somit die Ausführungsrechte eines Anwendungsfalls aus den Methodenaufrufen innerhalb des Anwendungsfalls berechnen. Da ein Anwendungsfall weiterhin einer oder mehrerer Aktivitäten zugeordnet ist, können wir somit auch die Ausführungsrechte einer Aktivität berechnen oder gegebenenfalls mit einer spezifizierten Ausführungsberechtigung (vgl. Abschnitt 6.3.3) vergleichen.

Grenzen der Berechenbarkeit von Ausführungsrechten

Wäre es allgemein möglich, die Ausführungsrechte von Aktivitäten und Anwendungsfällen aus den Zugriffsrechten auf Datenobjekten und Systemfunktionen zu berechnen, so wäre eine Spezifikation der Ausführungsrechte unnötig. Die Berechnung der Ausführungsrechte ist jedoch nicht immer möglich, wie die folgenden Einschränkungen zeigen:

- Die Objekt-Typen, auf denen innerhalb des Anwendungsfalls operiert wird, müssen vorab bekannt sein. Werden Objekt-Typen, auf denen ein Akteur innerhalb eines Anwendungsfalls operiert, erst zur Laufzeit bestimmt, so können auch die Zugriffsrechte erst zur Laufzeit bestimmt werden. Somit ist es nicht möglich, das Ausführungsrecht für den Anwendungsfall vor Ausführung des Anwendungsfalls zu bestimmen, da dieser andernfalls schon vorab zur Bestimmung des Ausführungsrechts durchlaufen werden müsste. Die Einschränkung der Berechnung der Berechtigung basiert in diesem Fall auf der gewählten *Granularität* des Anwendungsfalls. Es wäre hier nur möglich, den Anwendungsfall in kleineren Einheiten aufzuteilen und für diese Einheiten die Benutzerrechte vorab zu bestimmen.

- Müssen in einem Anwendungsfall spezielle Prinzipien (siehe Abschnitt 6.3.3) eingehalten werden, wie beispielsweise das *4-Eyes-Principle*, so können die Ausführungsrechte nicht aus den Zugriffsrechten berechnet werden.
- Durch die Kombination von Zugriffen innerhalb eines Ablaufs ändert sich der Wert der Information. Während zum Beispiel einzelne Zugriffe auf einzelne Objekte erlaubt sind, lässt ein Zugriff auf alle Objekte einer Klasse Schlussfolgerungen zu. Um derartige Schlussfolgerungen zu vermeiden, müssen Anwendungsfälle, die den Wert der Information erhöhen, durch eigens definierte Ausführungsrechte geschützt werden.

Wie insbesondere die zuletzt genannte Einschränkung zeigt, kann es notwendig oder sinnvoll sein, dass für einzelne Anwendungsfälle bzw. Aktivitäten die Benutzerrechte nicht berechnet, sondern spezifiziert werden. Hierbei können zusätzliche Bedingungen spezifiziert werden, sodass der Benutzerzugriff noch weiter eingeschränkt wird.

Weiterhin ist es auch möglich, die Benutzerrechte für Anwendungsfälle zu spezifizieren und für Datenobjekte zu berechnen. Innerhalb eines Ablaufs wird dann für alle Datenzugriffe die Berechtigung des Anwendungsfalls übernommen. Werden dieselben Daten von verschiedenen Anwendungsfällen gelesen und manipuliert, so errechnen sich die Zugriffsrechte aus der Kombination der Ausführungsrechte der Anwendungsfälle, die auf die Daten zugreifen. Da wir jedoch in unserem Ansatz davon ausgehen, dass die Zugriffsrechte spezifiziert werden, verfolgen wir innerhalb dieser Arbeit diesen Ansatz nicht weiter.

Berechnung der Ausführungsrechte

Sei einer Aktivität ein Anwendungsfall zugeordnet und in diesem Anwendungsfall seien ein exemplarischer Ablauf sowie Ablaufvarianten definiert. Dieser exemplarische Ablauf beschreibt eine Abfolge von Methodenaufrufen, wobei Methoden aufgerufen werden, die sich entweder in weitere Methodenaufrufe zerlegen lassen oder nicht weiter zerlegt werden können. Im letztgenannten Fall sprechen wir von *elementaren Methoden* m_E.

Gehen wir davon aus, dass zu den elementaren Methoden m_E Benutzerrechte spezifiziert sind, so können wir aus diesen Benutzerrechten die Benutzerrechte der aufrufenden, zerlegbaren Methode m_Z bestimmen. Ermitteln wir alle Benutzerrechte der teilbaren Methodenaufrufe derart, so können wir entgegengesetzt der Richtung der Methodenaufrufe die Benutzerrechte der teilbaren Methodenaufrufe und somit auch des gesamten Ablaufs ermitteln.

Sei $\mathcal{M}$ die Menge von Methoden, die aufgerufen werden können. Die Elemente von $\mathcal{M}$ sind Zustandsfunktionen $m_1, \ldots, m_n$. Über dieser Menge von Methoden führen wir eine zweistellige Relation $\mathcal{R}^P$ wie folgt ein:

$$\begin{aligned} &\mathcal{R}^P \subseteq \mathcal{M} \times \mathcal{M} \\ &(m_x, m_y) \in \mathcal{R}^P \ : \ m_y \text{ liefert Permission für } m_x, \end{aligned}$$

falls es im Ablauf eine Folge von Methodenaufrufen $m_0, \ldots, m_n$ mit $n \geq 1$ gibt und es gilt:

$$\begin{aligned} m_1 &= m_x \\ m_n &= m_y \\ m_x &= m_2(m_3(m_4(\ldots(m_{n-1}(m_y()))))) \end{aligned}$$

Ein Element (m_x, m_y) ist genau dann in der Relation $\mathcal{R}^P$ enthalten, wenn die Methode m_y für die Methode m_x eine Berechtigung liefert. Eine Methode liefert genau dann eine Berechtigung, wenn sie durch eine Folge von $0 \ldots n$ weiteren Methodenaufrufen aufgerufen wird.

Die Relation $\mathcal{R}^P$ liefert uns die *transitive Hülle* [Bro98b, OW96] über den Methodenaufrufen zu einer gegebenen Methode m, d.h., in ihr sind alle im Ablauf der Methode m erreichbaren Methodenaufrufe in der Form "Methode m_1 ruft Methode m_2 auf" enthalten.

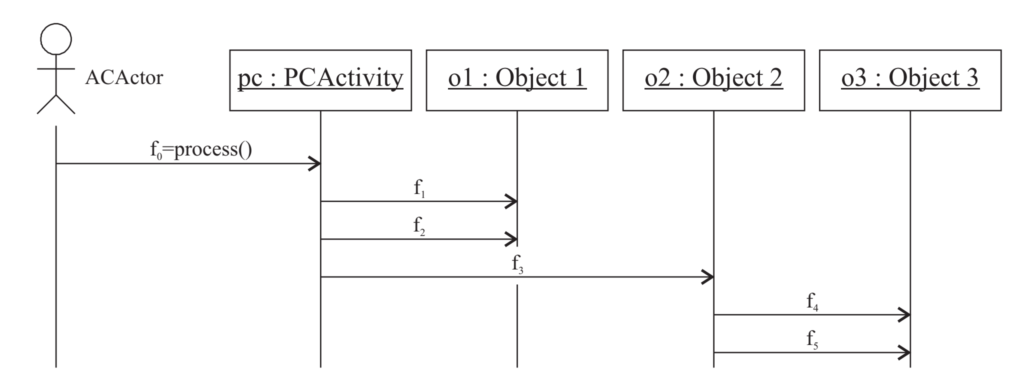


Abbildung 7.10: Exemplarischer Funktionsablauf

Als Beispiel betrachten wir den gegebenen Ablauf aus Abbildung 7.10 mit den Methoden $f_0, \ldots, f_5$. In der Relation $\mathcal{R}^P$ notieren wir alle Methodenaufrufe, die erreichbar sind und somit für die Ermittlung der Berechtigungen herangezogen werden können:

$$\mathcal{R}^P = \{(f_0, f_1), (f_0, f_2), (f_0, f_3), (f_3, f_4), (f_3, f_5), (f_0, f_4), (f_0, f_5)\}$$

Die ersten fünf Elemente sind die im Beispielablauf eingetragenen Methodenaufrufe. Durch die Transitivität sind von der Methode f_0 auch diejenigen Methoden erreichbar, die über die Methode f_3 aufgerufen werden.

Nachdem wir die Menge der Methodenaufrufe in der Menge $\mathcal{R}^P$ festgelegt haben, ist aus dieser Menge die Berechtigung des Ablaufs der Aktivität bzw. des Anwendungsfalls zu berechnen. Wir spezifizieren die Rechte für eine Methode m, welche den Ablauf des Anwendungsfalls und somit der Aktivität widerspiegelt (im Falle einer *1:1*-Zuordnung).

Da zur Ausführung des Anwendungsfalls bzw. der Aktivität die Benutzerrechte von allen im Anwendungsfall aufgerufenen Methoden erfüllt sein müssen, können wir die Berechtigung eines Anwendungsfalls mithilfe der *logischen UND-Verknüpfung* berechnen:

$$perm_{\,C,m}\,(ACActor, C, T_1, \ldots, T_n) = \bigwedge_{\forall\,(m_x,m_y)\,\in\,\mathcal{R}^P} perm\,(m_x, m_y) \quad \text{mit}$$

$$perm\left(m_x(ACActor, \ldots), m_y(ACActor, C, \ldots)\right) = \begin{cases} perm_{C,m_y}(ACActor, C, T_1, \ldots, T_n) \\ \qquad \text{falls } m_y \text{ elementare Methode,} \\ \bigwedge\limits_{i=1}^{n} perm(m_x, m_i) \text{ sonst.} \end{cases}$$

wobei die m_i in $\bigwedge\limits_{i=1}^{n} perm(m_x, m_i)$ die n im Ablauf aufgerufenen Submethoden der Methode m_x sind.

Durch die vorgestellte Berechnungsweise wird aus den Berechtigungen zu den elementaren Methoden die Berechtigung der Methode m der Klasse C unter Betrachtung des vorgestellten Ablaufs ermittelt.

Sind nun in einem Anwendungsfall mehrere, genauer j, Ablaufvarianten spezifiziert, so sind zu allen Ablaufvarianten und den dazugehörigen Ablaufmethoden m_j die Berechtigungen $perm_{C,m_j}$ nach dem oben gezeigten Verfahren zu berechnen. Durch die erneute *logische UND-Verknüpfung* der ermittelten Berechtigungen der Abläufe errechnet sich die Menge der Berechtigungen, die ein Akteur benötigt, sodass er alle Zweige der Methode m ausführen kann, wie folgt:

$$perm_{C,m} = \bigwedge_{\mu=1}^{j} perm_{C,m_\mu}$$

Nach diesem Berechnungsverfahren errechnen wir das *Minimum* der Berechtigungen, die ein Akteur benötigt, um alle Ablaufvarianten eines Anwendungsfalles ausführen zu können.

Betrachten wir im Folgenden ein einfaches Beispiel, bei dem wir von einem Objekt zur Stundenbuchung (*Accounting*) eine Kopie anlegen wollen. Der Ablauf hierzu ist in Abbildung 7.11

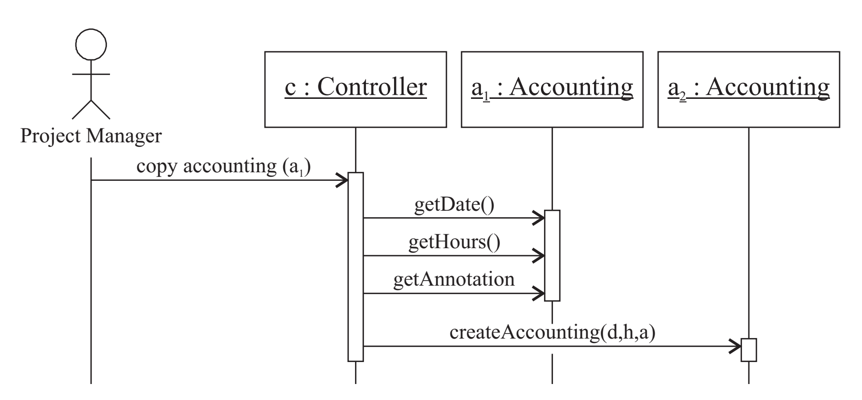


Abbildung 7.11: Funktionsablauf des TimeTool-Anwendungsfalls Copy Accounting

dargestellt. Für die elementaren Methoden des Ablaufs sind folgende Zugriffsrechte für den Akteur *Project Manager* spezifiziert (vgl. Abschnitt 4.4):

- Für die Lesezugriffe auf die Felder des zu kopierenden Buchungsobjekts ist folgendes Benutzerrecht gegeben (hier am konkreten Beispiel der getAccountingDate-Methode):

$$\begin{aligned}&\forall\, pm : ACProjectManager\ .\ \forall\, a1 : Accounting\ .\\&a1.activity.project.projectmanager = pm.rolerep\,()\\&\quad\Rightarrow\ perm_{\,Accounting,\,getDate}(pm, a1)\end{aligned}$$

  Die Methoden *getHours()* und *getDate()* besitzen dasselbe Zugriffsrecht als die Methode *getDate()*.

- Für die createAccounting-Methode wurde folgendes Benutzerrecht spezifiziert:

$$\forall pm : ACProjectManager \,.\, \forall ac : Activity \,.\, \forall u : User \,.\\ ac.project.projectmanager = pm.rolerep\,()\\ \Rightarrow\; perm_{\,Accounting,\,create}(pm, ac, u)$$

Aus den Zugriffsrechten zu den Elementarmethoden ergibt sich folgendes Ausführungsrecht für die Methode *copyAccounting* der Klasse *Accounting*:

$$\forall pm : ACProjectManager \,.\, \forall\, a1 : Accounting \,.\, \forall\, ac : Activity \,.\, \forall\, u : User\\ a1.activity.project.projectmanager = pm.rolerep\,() \,\wedge\\ ac.project.projectmanager = pm.rolerep\,()\\ \Rightarrow\; perm_{\,Accounting,\,copyAccounting}(pm, a1, ac, u)$$

Die Parameter *ac* und *u* für die zugeordnete Aktivität und den zugeordneten User des Objekts a_2 können bei der Berechnung der Berechtigung aus dem Objekt a_1 entnommen werden.

Bei der gezeigten Berechnungsvorschrift mittels der transitiven Hülle müssen zur Ermittlung des Benutzerrechts des Ablaufs auch alle Berechtigungen von teilbaren Methodenaufrufen m_Z bestimmt werden. Da die zugeordneten Berechtigungen aber keine zusätzliche Information für die Ermittlung der Berechtigung des Ablaufs ermitteln, kann die Ermittlung der Berechtigung noch vereinfacht und effizienter gestaltet werden: Mittels einer Tiefensuche im Ablauf sind alle elementaren Methoden m_E zu suchen. Die *logische UND-Verknüpfung* der Zugriffsrechte der elementaren Methoden ergibt dann genau die Berechtigung des Ablaufs. Auf die detaillierte Berechnung mittels Tiefensuche gehen wir nicht näher ein.

In unserem Ansatz sind wir bis jetzt immer davon ausgegangen, dass die Benutzerrechte basierend auf den Berechtigungen von elementaren Methoden m_E berechnen können. Es wäre jedoch möglich, dass für eine im Ablauf integrierte, teilbare Methode m_Z ein eigenes Ausführungsrecht spezifiziert wurde. In diesem Fall betrachten wir die ursprünglich teilbare Methode m_Z wie eine elementare Methode m_E mit zugeordnetem Benutzerrecht. Somit werden bei der Ermittlung des Benutzerrechts des Ablaufs die Submethoden der ursprünglich teilbaren Methode m_Z nicht mehr zur Berechnung herangezogen, sondern durch das vorgegebene Ausführungsrecht ersetzt. An der Ermittlung der Berechtigung des Ablaufs müssen ansonsten keine weiteren Änderungen vorgenommen werden.

Die explizite Angabe eines Ausführungsrechts für einen Teilablauf spielt insbesondere dann eine Rolle, wenn im Teilablauf eine Steigerung des Werts der Information erfolgt (siehe hierzu "Grenzen der Berechenbarkeit von Ausführungsrechten" weiter oben in diesem Abschnitt). Anstelle der zu berechnenden Berechtigung wird die zumeist stärker beschränkende, explizit spezifizierte Berechtigung für die weitere Ermittlung verwendet.

7.5 Der Prozess der Anwendungsfallmodellierung zugriffssicherer Systeme

In diesem Kapitel haben wir bereits einzelne Aktivitäten zur Erarbeitung und Verfeinerung von Aspekten der Zugriffssicherheit innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme vorgestellt. Im Folgenden zeigen wir die Eingliederung dieser Aktivitäten in den Prozess (Abschnitt 7.5.1) sowie die erneute Anwendung des Sicherheitsmikroprozesses (Abschnitt 7.5.2).

7.5.1 Prozessmuster 7.1: Anwendungsfallmodellierung zugriffssicherer Systeme

Kurzbeschreibung

Kernziele der Anwendungsfallmodellierung sind die Transformation der im System umzusetzenden Geschäftsprozesse in Anwendungsfälle und die Erweiterung des Domänenmodells. Für eine durchgängige und sukzessive Entwicklung der Zugriffssicherheit sind die bereits ermittelten Schutzziele an die neuen und verfeinerten Modelle im System anzupassen und ebenfalls zu verfeinern.

Um zu vermeiden, dass neu hinzugefügte oder geänderte Systemanforderungen keiner Sicherheitsanalyse unterzogen werden, müssen auch wie bei der Geschäftsprozessmodellierung zugriffssicherer Systeme Schutzziele sowie Bedrohungen, Risiken und Maßnahmen ermittelt werden.

Bei der Ablaufmodellierung ist zudem das Schutzziel Authentifikation zu betrachten, sodass geeignete Zeitpunkte und Techniken zur Authentifikation und Autorisierung ermittelt werden können.

Problem

Bei der Anwendungsfallmodellierung werden aus den Geschäftsprozessen und dem Domänenmodell die Anforderungen an das System abgeleitet. Struktur und Verhalten werden dabei verfeinert. Bekannte Verfahren zur Entwicklung der Systemanforderungen [JBR99, Kru00, DW98] beschränken sich hier auf die Modellierung der Funktionalität, die Transformation von nichtfunktionalen Anforderungen, wie zum Beispiel die Zugriffssicherheit, bleibt außen vor.

Zu Beginn der Anwendungsfallmodellierung ist festzulegen, welche von den ermittelten Geschäftsprozessen in dem zu entwickelnden System umgesetzt werden. Dadurch, dass nun die nicht zu automatisierenden Abläufe entfallen, sind die entwickelten funktionalen Modelle nicht mehr konsistent. Da wir die Anforderungen der Zugriffssicherheit gemeinsam mit der Funktionalität ermitteln, führen Inkonsistenzen und Lücken in den funktionalen Modellen und Beschreibungen zu Lücken und Inkonsistenzen in den Sicherheitsaspekten.

Im Geschäftsprozessmodell sind die Abläufe durch eine Folge von Aktivitäten beschrieben. Die Aktivitäten sind in einzelne Anwendungsfälle zu verfeinern, welche die internen Abläufe der Aktivitäten beschreiben. Eine Aktivität kann jedoch nicht immer genau einem Anwendungsfall zugeordnet werden. Ein Anwendungsfall stellt immer einen zusammenhängenden und vollständigen Ablauf dar, während die Granularität von Aktivitäten bei der Geschäftsprozessmodellierung nicht festgelegt ist. Veränderungen an der Aufteilung der Funktionalität im Rahmen der Bestimmung der Anwendungsfälle führen zu Inkonsistenzen und Lücken in den Beschreibungen von Aspekten der Zugriffssicherheit.

Für die Ausführung von sicherheitskritischen Funktionen müssen sich die Benutzer am System anmelden und die Benutzerrechte sind vor dem Zugriff auf Daten und vor der Ausführung von Funktionen abzuprüfen. Bei der Beschreibung des Ablaufverhaltens sind Aktionen und Techniken der Identitätsprüfung in die Entwicklung mit einzubeziehen.

Änderungen am Domänenmodell ergeben sich nicht nur aus der Festlegung der zu automatisierenden Prozesse. Im Rahmen der Modellierung der Anwendungsfälle wird das Datenmodell

mit weiteren Typen von Nachrichten, Schnittstellenobjekten, Vorgangsklassen, etc. erweitert und die bestehenden Klassen werden optimiert, d.h., Vererbungen, Assoziationen und Attribute, etc. werden angepasst. Diese Änderungen sind bei der Modellierung der Aspekte der Zugriffssicherheit ebenfalls zu betrachten.

Lösung

Für eine durchgängige und sukzessive Entwicklung der Anforderungen an die Zugriffssicherheit sind die bereits erarbeiteten Sicherheitsaspekte aus der Geschäftsprozessmodellierung zugriffssicherer Systeme in die verfeinerte Struktur und in das verfeinerte Verhalten der Anwendungsfallmodellierung zu übertragen, ergänzen und verfeinern.

Durch die Festlegung der Anforderungen des zu entwickelnden Systems sind diejenigen Teile aus den Geschäftsabläufen zu entfernen, die nicht automatisiert werden. Neben den Abläufen sind auch das Domänenmodell und die Akteure anzupassen. Veränderte Attribute und Assoziationen im Domänenmodell erfordern eine Überprüfung der Schutzziele und die Akteure sind für eine Rechtespezifikation gegebenenfalls neu hierarchisch anzuordnen. Durch die unterschiedliche Zuordnung von Aktivitäten zu Anwendungsfällen können die bereits im Verhalten spezifizierten Schutzziele und Benutzerrechte nicht immer direkt, d.h., 1:1, auf die Anwendungsfälle übernommen werden. Schutzziele, Bedrohungen, Risiken und Maßnahmen müssen gegebenenfalls für die Anwendungsfälle erneut geprüft und ermittelt werden.

In der Geschäftsprozessmodellierung zugriffssicherer Systeme wurde das Schutzziel Authentifikation nur für Flussobjekte untersucht. Für eine vollständige Beschreibung der Abläufe in den Anwendungsfällen ist es jedoch notwendig, die Aktivitäten der Authentifikationen und Autorisierungen zu ermitteln sowie die angewendeten Techniken zur Authentifikation zu beschreiben.

Abbildung 7.12 gibt einen Überblick über die Aktivitäten der Anwendungsfallmodellierung zugriffssicherer Systeme. Die Aktivitäten zur Modellierung der Zugriffssicherheit sind dabei grau hinterlegt.

Zur Modellierung von Authentifikation und Autorisierung sind in den Geschäftsprozessen Sitzungen (engl. *Sessions*) herauszuarbeiten und daraus Aktivitäten zur Prüfung der Authentifikation und Autorisierung zu ermitteln. Die anzuwendenden Authentifikationstechniken und die damit verbundenen Abläufe sind in eigenen Sicherheitsanwendungsfällen zu beschreiben.

Schutzziele, Bedrohungen, Risiken und Maßnahmen sind für Änderungen am Domänenmodell sowie für Anwendungsfälle, die nicht 1:1 zu einer Aktivität zugeordnet werden können, neu zu bestimmen. Mithilfe des Sicherheitsmikroprozesses aus Abschnitt 5.3.3 sind die Modelle einer erneuten Ermittlung zu unterziehen. Die Ergebnisse des Sicherheitsmikroprozesses sind auch für die Anpassung der Benutzerrechte heranzuziehen. Insbesondere müssen auch für Anwendungsfälle, die nicht direkt, d.h., nicht 1:1, aus Aktivitäten transformiert werden können, die Benutzerrechte spezifiziert und im Benutzerrechtemodell aktualisiert werden.



Abbildung 7.12: Die Aktivitäten der Anwendungsfallmodellierung zugriffssicherer Systeme

Aktivitäten

Innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme sind bei der Prozessausführung die folgenden Aktivitäten nach der in der Lösung und in der Struktur skizzierten Vorgehensweise auszuführen. Aus Gründen der Vollständigkeit wurden bei den Aktivitäten auch diejenigen aus der allgemeinen Anwendungsfallmodellierung mit aufgenommen und umrissen (vgl. Abschnitt 7.1.2).

- *Zu automatisierende Prozesse festlegen*
  In der Geschäftsprozessmodellierung wurden die Prozesse des Unternehmens bzw. der Organisation in Abläufen modelliert. Zu Beginn der Anwendungsfallmodellierung muss nun gemeinsam mit den Auftraggebern festgelegt werden, welche Prozesse in dem zu entwickelnden System umgesetzt werden. Im Rahmen einer iterativen Entwicklung können hier auch die Inkremente festgelegt werden.

- *Anpassung der Prozesse*
  Bedingt durch die Festlegung der automatisierbaren Prozesse ist das Geschäftsprozessmodell anzupassen. Sowohl gesamte, nicht automatisierbare Abläufe sowie einzelne nicht umzusetzende manuelle Aktivitäten sind aus dem Modell zu entfernen. Insbesondere die Entfernung von einzelnen Aktivitäten erfordert die Überarbeitung der Abläufe.

- *Anpassung des Domänenmodells*
  Das Domänenmodell muss an die im System umzusetzenden Abläufe angepasst werden. Klassen zu Daten, die aufgrund der Streichung von Geschäftsabläufen und Aktivitäten im Domänenmodell nicht gespeichert werden müssen, sind zu entfernen. Klassen, Beziehungen und Attribute des Domänenmodells sind nach der Entfernung von unnötigen Elementen anzupassen.

- *Anpassung des Akteurmodells*
  Die Festlegung der automatisierbaren Abläufe erfordert weiterhin eine Anpassung des Akteurmodells. Durch das Entfernen von Geschäftsabläufen sind gegebenenfalls auch Akteure aus dem Akteurmodell zu entfernen. Da innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme die Akteure in Bezug auf die Spezifikation der Benutzerrechte hierarchisch angeordnet wurden (vgl. Abschnitt 6.3.1), ist die Hierarchie anzupassen, d.h., Lücken in der Hierarchie sind zu schließen oder die Hierarchie ist neu anzuordnen. Da durch die Festlegung der im System umzusetzenden Funktionalität auch die Grenzen des Systems sowie externe Softwaresysteme und Hardware definiert werden können, sind auch die externen Systeme als Akteure im Akteurmodell aufzunehmen.

- *Modellierung von Sitzungen*
  Bei zugriffssicheren Systemen müssen sich Anwender zumeist nicht vor der Ausführung von einzelnen sicherheitskritischen Systemen am System authentifizieren, sondern zu Beginn einer Sitzung (mit Ausnahme von einigen Web-Anwendungen und Geldautomaten). In Hinblick auf die Trennung von Authentifikation und Autorisierung sind innerhalb der modellierten Abläufe Sitzungen zu ermitteln, innerhalb derer ein Akteur im System authentifiziert ist und Aktivitäten ausführen kann. Die Modellierung von Sitzungen ist Vorstufe für die Analyse von Authentifikation und Autorisierung. Die Modellierung von Sitzungen wird im Abschnitt 7.2.2 diskutiert.

- *Analyse der Anwendungsfälle*
  Basierend auf den Aktivitäten der Geschäftsprozesse ist das Verhalten in Anwendungsfällen zu verfeinern. Im Rahmen der Analyse der Anwendungsfälle sind diese zu ermitteln, jedoch noch nicht auszuarbeiten. Wie wir bereits im Abschnitt 7.3.1 beschrieben haben, werden Aktivitäten nicht immer direkt, d.h., 1:1, Anwendungsfällen zugeordnet, sondern *m:n*, d.h., ein oder mehrere Aktivitäten werden einem oder mehreren Anwendungsfällen zugeordnet.

- *Analyse von Authentifikation und Autorisierung*
  Im Rahmen der Modellierung von Sitzungen (siehe oben) werden in den Abläufen diejenigen Aktivitäten, für die ein Akteur authentifiziert sein muss, gekennzeichnet. Da jedoch nicht für jede Aktivität die Authentifikation, d.h., die Überprüfung der Echtheit und Glaubwürdigkeit des Akteurs, durchzuführen ist, müssen eigene Aktivitäten zur Prüfung der Authentifikation eingeführt werden. Bei den annotierten Aktivitäten ist über eine Rechteabprüfung (Autorisierung) oder die Durchführung einer Authentifikation zu entscheiden. Die Verfeinerung der Authentifikation und Sitzungen wird in Abschnitt 7.2.4 diskutiert.

- *Priorisierung der Anwendungsfälle*
  Für eine inkrementelle Entwicklung oder beispielsweise für den Bau eines funktionalen Prototypen muss festgelegt werden, in welcher Reihenfolge die Anwendungsfälle umzusetzen sind.

- *Ausarbeitung der Anwendungsfälle*
  Die in der Analyse der Anwendungsfälle (siehe oben) ermittelten Anwendungsfälle sind zu entwerfen und nach der in [Bre01] vorgestellten Struktur zu beschreiben. Für jeden Anwendungsfall sind beteiligte Akteure, Ein- und Ausgabedaten, Modifikationen am Anwendungskern, Ablaufbeschreibung sowie Varianten des Ablaufs zu beschreiben.

- *Entwicklung der Sicherheitsanwendungsfälle*
  Gemeinsam mit der Ausarbeitung der Anwendungsfälle sind für sicherheitskritische Funktionen die Anwendungsfälle zu erweitern und für Sicherheitsgrundfunktionen Sicherheitsanwendungsfälle zu ergänzen (siehe Abschnitt 7.3). Entspricht ein sicherheitskritischer Anwendungsfall genau einer Aktivität, können die ermittelten Schutzziele, Bedrohungen, Risiken, Maßnahmen sowie Benutzerechte aus der Geschäftsprozessmodellierung auf den Anwendungsfall übertragen werden. In den sonstigen Fällen ist der Anwendungsfall für eine erneute Ermittlung der Bedrohungen, Risiken und Maßnahmen sowie für eine Spezifikation der Benutzerrechte vorzumerken. Die Ermittlung der Bedrohungen, Risiken und Maßnahmen erfolgt durch die Anwendung des Sicherheitsmikroprozesses.

- *Ergänzung des Domänenmodells*
  Bei der Entwicklung der Anwendungsfälle werden die Ein- und Ausgaben sowie Änderungen am Anwendungskern spezifiziert. Diese Änderungen sind in das Domänenmodell zu übernehmen. Änderungen am Anwendungskern können etwa Umstrukturierungen, das Modifizieren oder Hinzufügen von Assoziationen und Attributen oder das Hinzufügen von neuen Klassen sein. Durch die Änderung von Klassen und Assoziationen sind die Klassen erneut einer Ermittlung der Schutzziele, Bedrohungen, Risiken und

Maßnahmen zu unterziehen (siehe Ausführung des Sicherheitsmikroprozesses). Anschließend sind die Benutzerrechte zu überprüfen.

- *Entwurf der UI-Prototypen*
  Im Rahmen der Anwendungsfallmodellierung sind Prototypen der Benutzerschnittstelle zu entwickeln.

- *Ausführung des Sicherheitsmikroprozesses*
  Bei der Ausführung des Sicherheitsmikroprozesses sind für jeden sicherheitskritischen Anwendungsfall, für den die Sicherheitseigenschaften nicht direkt aus der zugeordneten Aktivität übernommen werden konnte, Bedrohungen, Risiken und Maßnahmen zu ermitteln. Ebenso muss der Sicherheitsmikroprozess auch auf modifizierte Entitäten des Domänenmodells angewendet werden. Einzelheiten zum Sicherheitsmikroprozess sind im Abschnitt 7.5.2 und im Prozessmuster zum Sicherheitsmikroprozess in Abschnitt 5.3.3 beschieben.

- *Modellierung der Ausführungsrechte*
  Für alle Anwendungsfälle, die nicht direkt, d.h., 1:1, aus Aktivitäten übertragen werden konnten, sind die Ausführungsrechte zu überprüfen oder gegebenenfalls neu zu spezifizieren. Durch die vorausgehende Ausführung des Sicherheitsmikroprozesses wurden in den betreffenden Anwendungsfällen bereits die Schutzziele für potenzielle Angriffe spezifiziert. Aus diesen Schutzzielen und den ermittelten Risiken, Bedrohungen und Maßnahmen sind die Ausführungsrechte abzuleiten. Die Entwicklung der Benutzerrechte im Rahmen der Anwendungsfallmodellierung ist in Abschnitt 7.4 beschrieben.

  Bei der Entwicklung der Anwendungsfälle wird der interne Ablauf des Anwendungsfalls spezifiziert. Liegen zu den im internen Ablauf aufgerufenen Submethoden Benutzerrechte in Form von Zugriffsrechten auf Objekten oder in Form von Ausführungsrechten auf Systemfunktionen vor, so können die Ausführungsrechte der Anwendungsfälle berechnet werden. Die Voraussetzungen für Benutzerrechteberechnung und die eigentliche Berechnung ist in Abschnitt 7.4.2 beschrieben.

  Innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme waren viele Einzelheiten der Abläufe noch offen, sodass die Ausführungsrechte vorab größtenteils textuell spezifiziert wurden. Im Rahmen der Spezifikation der Anwendungsfälle wurde das Verhalten angepasst und detailliert beschrieben. Gegen Ende der Anwendungsfallmodellierung zugriffssicherer Systeme sind die Abläufe bis auf einige wenige Ausnahmen festgelegt und in geeignete Einheiten aufgeteilt. In diesem Stadium können die anfänglich nur textuell spezifizierten Benutzerrechte prädikativ spezifiziert und verfeinert werden, wie dies im Kapitel 4 vorgestellt wurde. Die Benutzerrechte zu den wenigen Abläufen, die noch kein *stabiles* Stadium erreicht haben, können weiter textuell beschrieben werden. Im Rahmen der Analyse zugriffssicherer Systeme (vgl. Kapitel 8) sind die Abläufe zu konkretisieren und die Benutzerrechte zu diesen Abläufen prädikativ zu spezifizieren.

  Die bereits spezifizierten Ausführungsrechte sind auch an das geänderte Akteurmodell anzupassen. Änderungen in der Hierarchie sowie neu hinzugefügte Akteure, beispielsweise für externe Systeme, sind bei der Spezifikation der Benutzerrechte heranzuziehen.

- *Modellierung der Zugriffsrechte*
  Modifikationen am Domänenmodell des zu entwickelnden Systems erfordern eine Anpassung der Zugriffsechte. Sowohl ergänzte Klassen als auch modifizierte Attribute, Asso-

ziationen oder Klassen erfordern eine erneute Überprüfung der Benutzerrechte. Zudem sind die Zugriffsrechte auch den Änderungen und Ergänzungen im Akteurmodell anzupassen.

Ebenso wie bei der Modellierung der Ausführungsrechte erreicht das Domänenmodell innerhalb der Anforderungsentwicklung mehr und mehr einen *stabilen* Zustand, sodass die anfänglich textuell spezifizierten Zugriffsrechte in Hinblick auf eine automatisierte Codegenerierung prädikativ zu spezifizieren und zu verfeinern sind.

Produktartefakte

Die folgende Auflistung beschreibt die Eingaben und Ausgaben zum Prozess der Anwendungsfallmodellierung zugriffssicherer Systeme. Um Sicherheitsaspekte erweiterte Produktartefakte der allgemeinen Anwendungsfallmodellierung sind gekennzeichnet.

Eingabe-Produktartefakte

- Business View Modell (aus der Startphase)
- Domänenmodell (mit Sicherheitsaspekten erweitert)
- Geschäftsprozessmodell (mit Sicherheitsaspekten erweitert)
- Bedrohungs- und Risikomodell
- Benutzerrechtemodell

Ausgabe-Produktartefakte

- Domänenmodell (mit Sicherheitsaspekten erweitert)
- Geschäftsprozessmodell (mit Sicherheitsaspekten erweitert)
- Bedrohungs- und Risikomodell (erweitert und verfeinert)
- Benutzerrechtemodell (erweitert und verfeinert)
- Anwendungsfallmodell (mit Sicherheitsaspekten erweitert)

Kontext

Dieses Prozessmuster kann nur im Rahmen einer Anforderungsspezifikation zugriffssicherer Systeme (vgl. Abschnitt 5.3.2), basierend auf einem objektorientierten Vorgehensmodell, ausgeführt werden. Für die Ausführung des Prozessmusters werden mit Schutzzielen annotierte Geschäftsprozesse und Domänenobjekte vorausgesetzt. Weiterhin müssen die Benutzerrechte in einer Benutzerrechtematrix spezifiziert werden. Der Fokus dieser Anwendungsfallmodellierung zugriffssicherer Systeme liegt auf einer Neuentwicklung eines betrieblichen Informationssystems. Aktivitäten zur Analyse bestehender Systeme sind kein Bestandteil dieses Prozessmusters.

Neben den Entwicklungsschritten sind auch projektübergreifend Managementaufgaben durchzuführen, auf die jedoch im Rahmen dieses Prozessmusters nicht näher eingegangen wird.

Struktur

Innerhalb der Lösung wurde bereits der Ablauf der Anwendungsfallmodellierung zugriffssicherer Systeme skizziert. Im Aktivitätsdiagramm in Abbildung 7.13 ist der verfeinerte Ablauf dargestellt. Zudem werden auch geeignete Iterationsmöglichkeiten aufgezeigt.

Bei der Entwicklung der Anwendungsfälle können die Anwendungsfälle in Inkrementen ausgearbeitet werden. Die Iterationsmöglichkeiten des Ablaufs der Anwendungsfallmodellierung

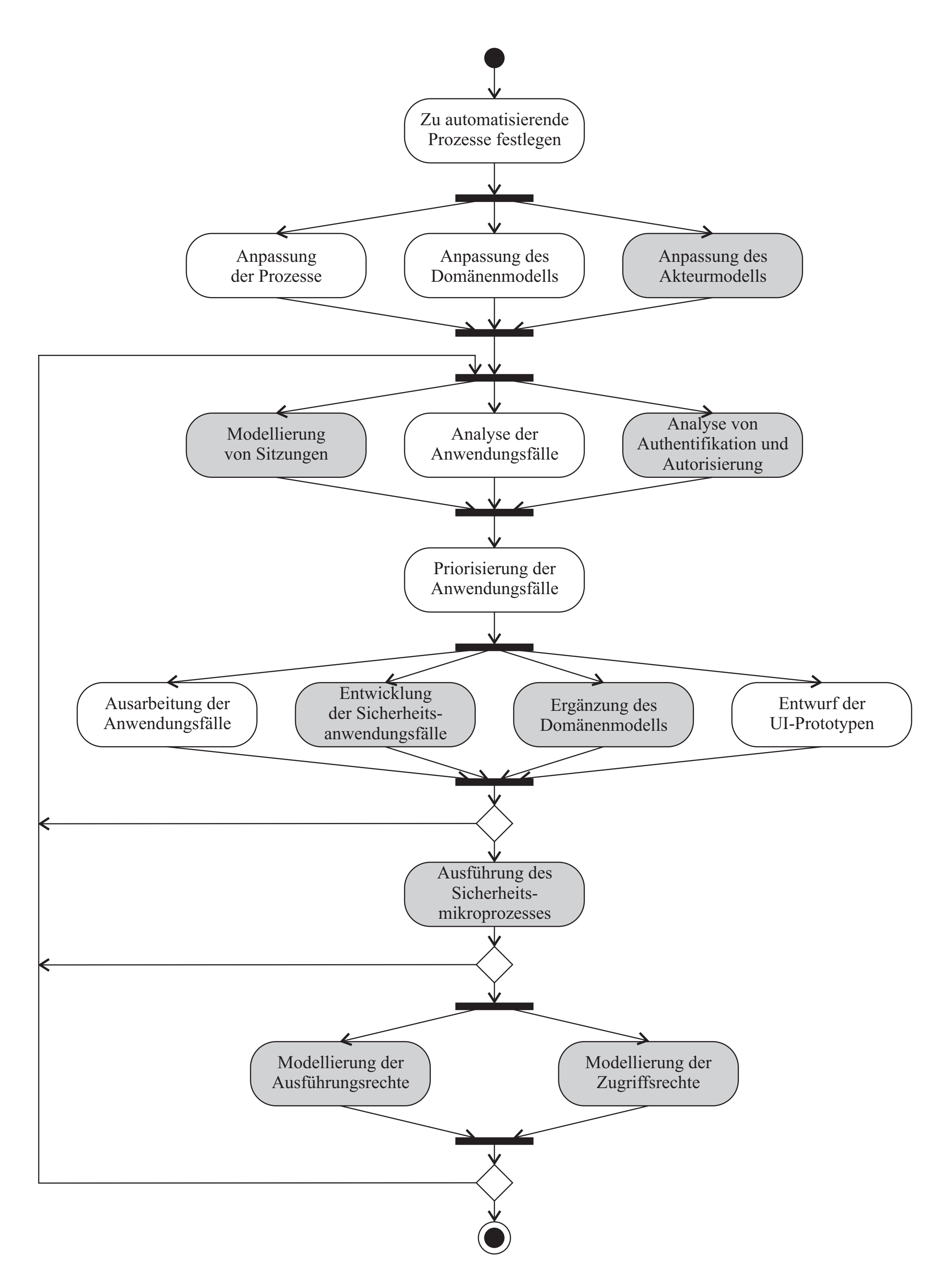


Abbildung 7.13: Der Ablauf der Anwendungsfallmodellierung zugriffssicherer Systeme

zugriffssicherer Systeme erlauben es, dass beispielsweise die Anwendungsfälle zu einem Geschäftsprozess vollständig ausgearbeitet werden. Anschließend steht es dem Entwickler offen, weitere Anwendungsfälle auszuarbeiten oder Bedrohungen und Risiken durch die Ausführung des Sicherheitsmikroprozesses zu ermitteln. Ebenso können die Berechtigungen jeweils nach Ermittlung der Schutzziele oder am Ende der Anwendungsfallmodellierung zugriffssicherer Systeme bestimmt werden.

Durch den vorgestellten Prozess wird die gemeinsame Entwicklung von Funktionalität und Sicherheit gefordert. Sobald Aspekte der Zugriffssicherheit betrachtet werden können, schreibt der Prozess die Behandlung von Sicherheitsaspekten vor. Insbesondere sind die Modellierung der Akteurhierarchie, die Modellierung von Sitzungen, Authentifikation und Autorisierung sowie die Entwicklung der Sicherheitsanwendungsfälle stark mit der funktionalen Entwicklung verzahnt. Schutzziele, Bedrohungen, Maßnahmen und Benutzerrechte müssen vor Abschluss der Anwendungsfallmodellierung zugriffssicherer Systeme ermittelt werden.

Vor- und Nachteile

Durch das vorgestellte Prozessmuster wird die frühzeitige Entwicklung von Sicherheit innerhalb der Phase der Anforderungsspezifikation zugriffssicherer Systeme weiter unterstützt. Durch die Betrachtung von Sicherheitsaspekten in allen Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme wird eine durchgehende Sicherheitsanalyse erreicht. Insbesondere wird durch den vorgestellten Prozess die geforderte sukzessive Entwicklung erreicht, da die Ergebnisse aus der Geschäftsprozessmodellierung zugriffssicherer Systeme in die Modellierung der Anwendungsfälle einfließen und dabei verfeinert und ergänzt werden.

Wie bereits in der Geschäftsprozessmodellierung zugriffssicherer Systeme begonnen, bleibt das Klassendiagramm frei von Benutzerrechten. Hierzu werden Zugriffs- und Ausführungsrechte im Benutzerrechtemodell, d.h., in einer Benutzerrechtematrix, spezifiziert. In dieser Benutzerrechtematrix können sowohl Zugriffs- als auch Ausführungsrechte einheitlich dargestellt werden.

Das Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme bietet, wie bereits erwähnt, eine Vielzahl von Iterationsmöglichkeiten (vgl. Struktur des Prozessmusters). Besonders für große Systeme spielt die iterative Entwicklung eine bedeutende Rolle. Eine entscheidende Rolle spielt dabei die Tatsache, dass die Entwicklung von Aspekten der Zugriffssicherheit nicht in den Hintergrund treten dürfen, d.h., dass Sicherheitsaspekte in einer eigenen Iteration am Ende ermittelt werden. Bei den vorgestellten Iterationsmöglichkeiten kann das System in kleineren Inkrementen entwickelt werden, Sicherheitsaspekte sind dabei in jeder Iteration durchgehend zu ermitteln.

Wie bereits bei der Geschäftsprozessmodellierung zugriffssicherer Systeme erwähnt, erfordert dieses Verfahren der integrierten Entwicklung von Sicherheitsaspekten zusätzlichen Aufwand. Besonders bei der Entwicklung der Anwendungsfälle müssen gegebenenfalls Schutzziele, Bedrohungen, Risiken und Maßnahmen aus der Geschäftsprozessmodellierung zugriffssicherer Systeme erneut überprüft werden. Dem ist jedoch entgegenzuhalten, dass es für die Entwicklung von zugriffssicheren Systemen nicht ausreicht, Sicherheitsaspekte in einem einzigen Durchgang zu ermitteln. Während bei der Entwicklung von funktionalen Anforderungen das Ziel klar vor Augen und direkt verfolgt werden kann, müssen bei der Entwicklung der Zugriffssicherheit alle potenziellen Angriffe und somit Verletzungsmöglichkeiten des Systems erfasst werden. Der zusätzliche Overhead in der Analyse ist somit gerechtfertigt.

In Beziehung stehende Prozessmuster

Übergeordneter Prozess

- Prozessmuster zur Anforderungsspezifikation zugriffssicherer Systeme (siehe Abschnitt 5.3.2)

Auszuführender Teilprozess

- Prozessmuster zum Sicherheitsmikroprozess (siehe Abschnitt 5.3.3)

Weitere referenzierte Prozessmuster

- Prozessmuster zur Geschäftsprozessmodellierung zugriffssicherer Systeme (siehe Abschnitt 6.4.1)
- Prozessmuster zur Analyse zugriffssicherer Systeme (siehe Abschnitt 8.4.1) ◇

7.5.2 Anwendung des Sicherheitsmikroprozesses

Bei der Beschreibung des Prozessmusters zur Anwendungsfallmodellierung zugriffssicherer Systeme haben wir bereits mehrfach die wiederholte Ausführung des Sicherheitsmikroprozesses erwähnt. Wir fassen hier nochmals kurz zusammen, von welchen Änderungen und Weiterentwicklungen der Modelle die erneute Anwendung des Sicherheitsmikroprozesses getrieben wird.

Folgende Änderungen am *Domänenmodell* sind bei der Ausführung des Mikroprozesses zu beachten:

- *Festlegung der zu automatisierbaren Geschäftsprozesse und Definition der Gesamtfunktionalität des zu entwickelnden Systems.* Durch diese Festlegung sind im Klassendiagramm Klassen, Attribute und Assoziationen für die Kommunikation mit angrenzenden Systemen einzuführen und überflüssige Klassen, Attribute und Assoziationen, die aufgrund der Festlegung der Systemgrenzen nicht mehr notwendig sind, zu entfernen. Die verbleibenden Assoziationen sind an die neue Struktur anzupassen.

- *Definition der Anwendungsfälle.* Die bei der Spezifikation der Anwendungsfälle ermittelten Ein- und Ausgabedaten sowie Modifikationen an den Klassen des Anwendungskerns sind im Domänenmodell zu integrieren.

- *Maßnahmen der Sicherheitsanalyse in der Geschäftsprozessmodellierung zugriffssicherer Systeme.* Wurden dabei Maßnahmen zur Abwehr von Bedrohungen getroffen, die sich auf die Struktur des Anwendungskerns auswirken, so sind diese Änderungen am Anwendungskern bei dieser erneuten Ausführung des Sicherheitsmikroprozesses zu beachten. Beispielsweise wurde in unserer *TimeTool*-Fallstudie zur Nichtabstreitbarkeit von Änderungen an Buchungen eine *Logging*-Klasse im Anwendungskern eingefügt. Schutzziele und Bedrohungen auf diese Logging-Klasse müssen spezifiziert werden.

Der Sicherheitsmikroprozess muss auf folgende Änderungen in den *Ablaufbeschreibungen* Einfluss nehmen:

- *Anpassung der Geschäftsprozesse an die Funktionalität des Systems.* Bedingt durch die Festlegung der Grenzen des Systems sind auch die Abläufe zu überarbeiten, d.h., manuelle Aktivitäten oder gesamte manuell ausführbare Geschäftsprozesse sind von der Beschreibung der Systemfunktionalität zu entfernen. Die zu automatisierenden Abläufe müssen an diese Änderungen angepasst werden und Aktivitäten zur Interaktion mit externen Systemen sind zu spezifizieren. Änderungen und Ergänzungen der Abläufe müssen bezüglich Aspekten der Zugriffssicherheit untersucht werden.
- *Übergang von Geschäftsprozessen zu Anwendungsfällen.* Wie in Abschnitt 7.3 beschrieben wurde, wird das Verhalten aus den Aktivitäten der Geschäftsprozesse in Anwendungsfallbeschreibungen transformiert. Gehen dabei aus einer Aktivität mehrere Anwendungsfälle hervor oder werden mehrere Aktivitäten in einen gemeinsamen Anwendungsfall transformiert, müssen die resultierenden Anwendungsfälle einer Sicherheitsanalyse unterzogen werden.
- *Einführung von Anwendungsfällen für Sicherheitsgrundfunktionen.* Zur Beschreibung der Funktionalität der Sicherheitsmechanismen sind Sicherheitsanwendungsfälle dem System hinzuzufügen. Für diese Anwendungsfälle sind ebenfalls Schutzziele, Bedrohungen, Risiken und Maßnahmen zu ermitteln.

Die eigentliche Ausführung des Sicherheitsmikroprozesses ist im Prozessmuster in Abschnitt 5.3.3 beschrieben. Die ermittelten und veränderten Bedrohungen, Risiken, Maßnahmen sowie die Überprüfungsergebnisse der Maßnahmen werden im Bedrohungs- und Risikomodell spezifiziert.

7.6 Produktartefakte der Anwendungsfallmodellierung zugriffssicherer Systeme

Abschließend betrachten wir die Produktartefakte zur Anwendungsfallmodellierung zugriffssicherer Systeme und gehen dabei auf Ergänzungen bezüglich der Modellierung der Zugriffssicherheit ein.

In Abbildung 7.14 sind die Produktartefakte aus dem Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme mit ihren Teilprodukten dargestellt. Dabei sind alle durch Sicherheitsaspekte veränderten oder hinzugefügten Teilprodukte grau hinterlegt. Veränderungen an den Produkten des Produktmodells innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme sind in der in Abschnitt 6.5 vorgestellten Art gekennzeichnet.

Die Produkte des *Bussiness View Modells* (Projektidee, Projektmission und globales Sicherheitsziel) werden bei der Anwendungsfallmodellierung zugriffssicherer Systeme betrachtet, jedoch nicht verändert, ebenso der Überblick über die Architektur, die globalen Regeln und allgemeine Ergänzungen aus dem *Geschäftsprozessmodell.* Alle diese Produkte sind Rahmenbedingungen für die Sicherheitsanalyse. Das Glossar wird mit neu eingeführten Begriffen ergänzt. Durch die Festlegung der Systemgrenzen sind die zu automatisierenden Geschäftsprozesse auszuwählen und die Abläufe gegebenenfalls mit weiteren Aktivitäten und Assoziationen zu ergänzen. Insbesondere sind hier Aktivitäten zur Authentifikation einzubringen und die Autorisierung festzulegen. Die Akteure werden ebenfalls an die Systemgrenzen angepasst. Dabei sind auch externe Systeme zu den Akteuren hinzuzufügen und hierarchisch anzuordnen.

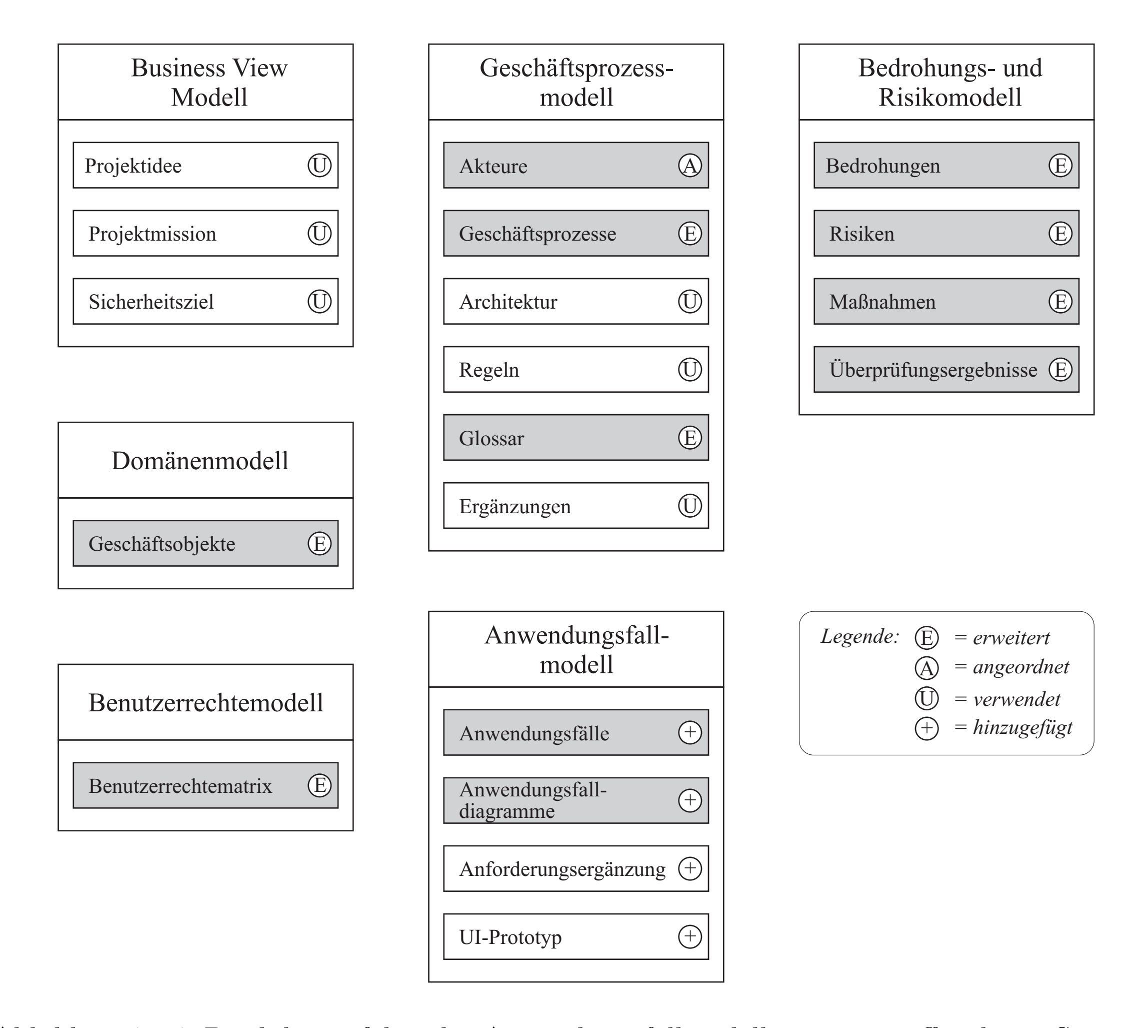


Abbildung 7.14: Produktartefakte der Anwendungsfallmodellierung zugriffssicherer Systeme

Die Geschäftsobjekte des *Domänenmodells* müssen auf die automatisierbaren Geschäftsprozesse zugeschnitten werden, die Klassen, Attribute und Assoziationen sind zu optimieren. Ein- und Ausgabetypen der Anwendungsfälle sowie Klassen für Schnittstellen zu externen Systemen werden aufgenommen. Für so genannte Sicherheitsgrundfunktionen sind Anwendungsfälle zu erstellen. Erfordern diese Sicherheitsanwendungsfälle spezielle Domänenobjekte sowie Ein- und Ausgabeobjekte, so müssen auch diese dem Domänenmodell hinzugefügt werden.

Im *Benutzerrechtemodell* sowie im *Bedrohungs- und Risikomodell* sind Änderungen und Ergänzungen der Berechtigungen sowie neue Bedrohungen, Risiken, Maßnahmen und Überprüfungen hinzuzufügen.

Im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme wird das *Anwendungsfallmodell* neu erstellt und zu den Produktartefakten hinzugefügt. Zentraler Bestandteil dieses Modells sind die strukturellen Beschreibungen der Anwendungsfälle mit Betrachtung der Zugriffssicherheit. Einen Überblick über die Anwendungsfälle des Systems und die Zuordnung zu den beteiligten Akteuren geben Anwendungsfalldiagramme, in die auch die Sicherheitsanwendungsfälle zur Beschreibung der Sicherheitsgrundfunktionen mit aufgenommen werden müssen. Zusätzliche Anforderungen, wie beispielsweise gesetzliche Regelungen, werden in der Anforderungsergänzung beschrieben, ein Entwurf der Benutzerschnittstelle im UI-Prototyp.

7.7 Zusammenfassung

Ziel der Anwendungsfallmodellierung zugriffssicherer Systeme ist es, die durchgängige und sukzessive Entwicklung von Aspekten der Zugriffssicherheit, die bereits bei der Analyse der Geschäftsprozesse begonnen wurde, fortzusetzen. Hierzu werden bei der Entwicklung von Struktur und Verhalten die bereits spezifizierten Anforderungen an die Zugriffssicherheit aufgegriffen und angepasst bzw. verfeinert.

Änderungen am Domänenmodell, die sich sowohl aus der Abgrenzung der Systemgrenzen, der detaillierten funktionalen Spezifikation sowie als Reaktion auf bereits ermittelte Maßnahmen zur Zugriffssicherheit ergeben, sind erneut einer Sicherheitsanalyse zu unterziehen. Durch die iterative Anwendung des Sicherheitsmikroprozesses werden für geänderte Attribute, Klassen und Assoziationen die Schutzziele, Bedrohungen und Risiken und Maßnahmen angepasst.

Die Festlegung der Systemgrenzen bewirkt eine Anpassung der Geschäftsabläufe. Sowohl diese Änderungen wie auch die Überführung der Aktivitäten der Geschäftsprozesse in Anwendungsfälle erfordern an den spezifizierten Abläufen eine erneute Überprüfung von Aspekten der Zugriffssicherheit. Mithilfe des Sicherheitsmikroprozesses können potenzielle Angriffsziele ermittelt und frühzeitig geeignete Maßnahmen zur Abwehr entwickelt werden.

Neben der Analyse der Anwendungsfälle ist die Modellierung des Schutzziels Authentifikation in der Anwendungsfallmodellierung zugriffssicherer Systeme zentral. Für die eigentliche Überprüfung der Echtheit und Glaubwürdigkeit der Akteure werden Aktivitäten zur Authentifikation in die Geschäftsprozesse aufgenommen. In Hinblick auf die Autorisierung der Akteure sind Vorbedingungen in die Anwendungsfälle aufzunehmen. Für die Modellierung des Schutzziels Authentifikation sind UML-Aktivitätsdiagramme um Beschreibungstechniken für Sitzungen und Authentifikation zu erweitern. Zur Beschreibung von Anforderungen der Zugriffssicherheit in Anwendungsfällen wird die strukturelle Beschreibung erweitert, für die Beschreibung von angewendeten Sicherheitsgrundfunktionen, wie beispielsweise die verwendeten Authentifikationstechniken, sind so genannte Sicherheitsanwendungsfälle zur Anforderungsspezifikation hinzuzufügen.

Durch die Spezifikation von Aspekten der Zugriffssicherheit bei der Entwicklung der Anwendungsfälle werden auch die Ziele der gemeinsamen Entwicklung von Funktionalität und Sicherheit sowie die lokale Spezifikation abgedeckt. Die Sicherheitsaspekte werden in kleinen Einheiten spezifiziert, sodass diese nicht erst nach Fertigstellung des Gesamtsystems systematisch aufgebrochen und analysiert werden müssen.

Der im Rahmen des Prozessmusters zur Anwendungsfallmodellierung zugriffssicherer Systeme vorgestellte Prozess sowie der Sicherheitsmikroprozess ermöglichen eine iterative Entwicklung der Funktionalität und der Aspekte der Zugriffssicherheit. Durch die gezeigte Methode ist die Entwicklung von Aspekten der Zugriffssicherheit fest in den Prozess verankert. In Hinblick auf eine durchgängige und sukzessive Entwicklung sowie eine vollständige Beschreibung der Prozessbausteine in Prozessmustern sind auch die Produktartefakte der Anwendungsfallmodellierung zugriffssicherer Systeme zentral. Die Produktartefakte wurden im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme bezüglich Informationen zur Beschreibung von Aspekten der Zugriffssicherheit untersucht.

Bezüglich der Änderungen an Struktur und Verhalten sowie der Verfeinerung des Verhaltens in Anwendungsfällen muss das Benutzerrechtemodell angepasst werden. Dies wird ebenfalls von

einer Anpassung des Akteurmodells getrieben, die sich aus der Festlegung der Systemgrenzen ergibt. Für Teile der Struktur- und Verhaltensbeschreibung, die innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme einen stabilen Zustand erreichen, sind die anfänglich textuell spezifizierten Benutzerrechten in Hinblick auf eine Codegenerierung Benutzerrechte prädikativ zu spezifizieren. Für diese prädikativ spezifizierte Benutzerrechte wurde unter Beschränkung der Allgemeinheit gezeigt, wie aus Zugriffsrechten Ausführungsrechte berechnet werden können.

8 Die Analyse zugriffssicherer Systeme

Im vorausgehenden Kapitel haben wir die Anwendungsfallmodellierung zugriffssicherer Systeme untersucht. Dabei wurde das Verhalten in Anwendungsfällen verfeinert und das Domänenmodell angepasst. Die Geschäftsabläufe sind vorab mit Aktivitäten zur Authentifikation ergänzt worden und in den Geschäftsprozessen spezifizierte Aspekte der Sicherheit sind in die Anwendungsfälle übertragen worden. Im Vordergrund stand die isolierte Betrachtung der Anwendungsfälle. Zur Beschreibung wurde die Sprache der Anwender und Auftraggeber verwendet und die Funktionalität war dabei in den Anwendungsfällen vollständig und intuitiv zu spezifizieren (vgl. [JBR99]).

Bei der Analyse sind die Systemanforderungen in eine für die Entwickler geeignete Form zu überführen. Im Rahmen der Anwendungsfallmodellierung wurden die Systemanforderungen in Anwendungsfällen spezifiziert. Für das weitere Softwaredesign sind diese Anforderungen auf Klassen und Subsysteme abzubilden. Diese Transformation wird getrieben durch die Anwendungsfälle und weitere nichtfunktionale Anforderungen. Die Analyse führt somit zu einem nahezu idealen Abbild des Systems (vgl. [Kru00]). Der Fokus der klassischen Analysephase [JBR99, Kru00] liegt auf den funktionalen Anforderungen. Einen Überblick über die Aktivitäten der Analyse geben wir in Abschnitt 8.1.

Bei den heutigen gängigen Methoden zur Analyse [Kru00, JBR99] spielen nichtfunktionale Anforderungen eine untergeordnete Rolle. [JBR99] führt beispielsweise eine eigene Aktivität zur Erfassung von *Special Requirements* ein, gibt jedoch kein schrittweises Verfahren zur Entwicklung und Verfeinerung dieser Anforderungen an. Auch für die schrittweise und sukzessive Analyse von Aspekten der Zugriffssicherheit als eine Ausprägung der nichtfunktionalen Anforderungen liefern gängige Entwicklungsmethoden keine Lösung. Anforderungen an die Sicherheit und weitere funktionale Anforderungen werden nur aufgenommen, jedoch beginnt die Ausarbeitung dieser Anforderungen erst in der Designphase.

Innerhalb der Analyse zugriffssicherer Systeme kann jedoch die in der Geschäftsprozessmodellierung begonnene und in der Anwendungsfallmodellierung fortgesetzte Analyse von Aspekten der Zugriffssicherheit schrittweise und sukzessive fortgesetzt werden. Maßnahmen, die zur Abwehr von Bedrohungen mit nicht vermeidbarem Risiko spezifiziert wurden, sind sowohl bei der Entwicklung des Analyseklassendiagramms als auch bei der Realisierung der Anwendungsfälle zu betrachten. Die innerhalb der Analyse gewählten Methoden und Techniken zur Abdeckung von Aspekten der Zugriffssicherheit erweitern und verfeinen die Modelle, bringen aber neue potenzielle Angriffsmöglichkeiten mit sich. Zur Ermittlung der Bedrohungen ist im Rahmen der Analyse zugriffssicherer Systeme für diese Erweiterungen eine Schutzzielanalyse nach dem bereits mehrfach eingesetzten Verfahren durchzuführen.

Für die schrittweise und sukzessive Verfeinerung des Verhaltens und der Aspekte der Zugriffssicherheit gehen wir in Abschnitt 8.2 auf die Realisierung der Anwendungsfälle innerhalb der Analysephase ein. Aus den Anwendungsfällen sind Interaktionen der Objekte sowie Abläufe zu

modellieren, die die bereits in den Anwendungsfällen spezifizierten Sicherheitsanforderungen umsetzen.

Abschnitt 8.3 beschäftigt sich mit der Auswahl und Integration von Sicherheitsprotokollen. Hier sind zur Realisierung von Anforderungen der Zugriffssicherheit gängige Sicherheitsprotokolle auszuwählen oder eigene Protokolle zu definieren.

In Abschnitt 8.4 stellen wir den Prozess zur Analyse zugriffssicherer Systeme in einem Prozessmuster vor. Dabei gehen wir auf die Aktivitäten zur Modellierung des Analyseklassendiagramms näher ein und beschreiben die Integration des Sicherheitsmikroprozesses in die Analyse zugriffssicherer Systeme. Abschließend betrachten wir in Abschnitt 8.5 die Produktartefakte der Analyse zugriffssicherer Systeme.

8.1 Grundlagen der Analyse

Bevor wir auf die Besonderheiten der Analyse zugriffssicherer Systeme eingehen, geben wir einen Überblick über Ziele, Aktivitäten und Produktartefakte der klassischen Analyse.

8.1.1 Motivation

In der Analyse werden die Systemanforderungen aus der Anwendungsfallmodellierung verfeinert und strukturiert. Ziele der Analyse sind, ein genaueres Verständnis der Anforderungen zu ermitteln und eine genaue Anforderungsbeschreibung zu erstellen. Die Beschreibung muss einfach umzusetzen sein und sie muss die Struktur des gesamten Systems, inklusive der Architektur, wiedergeben (siehe [JBR99]).

Bis jetzt wurden die Systemanforderungen größtenteils textuell oder in einfachen Klassen- und Aktivitätsdiagrammen spezifiziert. Bei der Analyse werden diese Beschreibungen in die Sprache der Entwickler umgesetzt und verfeinert. Hierbei sind formale oder semiformale Beschreibungen mit mehr Aussagekraft gegenüber textuellen Beschreibungen vorzuziehen.

Im Analysemodell werden die Anforderungen neu strukturiert, sodass diese besser verstanden werden können, in Hinblick auf das Design geeigneter dargestellt sind und einfacher abgeändert werden können. Beispielsweise sind Abläufänderungen in einem Sequenzdiagramm verständlicher und ersichtlicher als Änderungen an einer textuellen Ablaufbeschreibung.

In den Anwendungsfällen werden Daten und Abläufe teilweise noch redundant spezifiziert, zum Beispiel werden in den textuellen Ablaufbeschreibungen die Varianten eigenständig beschrieben. Synergien mit anderen Anwendungsfällen werden zu diesem Zeitpunkt nicht herausgearbeitet, ebenso auch bei der Ermittlung des Domänenmodells. Weiterhin werden bei der Analyse der Anwendungsfälle, getrieben durch die eigenständige Beschreibung, auch Inkonsistenzen nicht erkannt. Im Rahmen der Analyse sind Redundanzen und Inkonsistenten zu erkennen und zu beheben.

Ein Analysemodell kann als erster Entwurf eines Designmodells betrachtet werden. Das Analysemodell stellt eine wichtige Vorarbeit zum Design und zur Implementierung dar.

8.1.2 Aktivitäten der Analyse

Zu Beginn der Analyse ist die Architektur des zu entwickelnden Systems zu untersuchen. Die Anwendungsfälle und Entitäten des Domänenmodells aus der Anwendungsfallmodellierung sind dabei in kleinere, handhabbarere Einheiten für die Analyse aufzuteilen. Die so genannten *Analyse-Packages* können beispielsweise nach funktionalen Anforderungen oder nach Aufgabenbereichen der Unternehmung bzw. Organisation aufgeteilt werden. Weitere Unterteilungen der Packages sind in [JBR99] beschrieben.

Aus dem Domänenmodell der Anwendungsfallmodellierung sind die Klassen des Anwendungskerns für das Analyseklassendiagramm zu erstellen. Grundsätzlich werden im Klassendiagramm der Analyse folgende drei Typen von Klassen unterschieden: Interaktionsklassen, Vorgangsklassen und Fachklassen (siehe [Bre01, JBR99]). Da innerhalb der Geschäftsprozess- und Anwendungsfallmodellierung noch keine Vorgangsklassen bestimmt wurden, sind die Klassen des Anwendungskerns in Fachklassen und Interaktionsklassen aufzuteilen. Dieses Klassendiagramm der Analyse wird bei der Realisierung der Anwendungsfälle ergänzt. Bereits ermittelte Schnittstellen zu externen Systemen werden im Klassendiagramm durch Interaktionsklassen ergänzt. Datentypen für die Ein- und Ausgaben zu Anwendungsfällen, die im Rahmen der Anwendungsfallmodellierung ermittelt wurden, werden als Fachklassen in das Analyseklassendiagramm transformiert.

Nach der Verfeinerung des Domänenmodells ist die *Realisierung der Anwendungsfälle* in so genannten *Szenarien* zu erarbeiten. Dazu sind die Klassen, zwischen denen die Interaktionen des Anwendungsfalls stattfinden, aus den Anwendungsfallbeschreibungen und dem Analyseklassenmodell zu ermitteln. Falls diese Klassen noch kein Bestandteil des Analyseklassendiagramms sind, sind sie zu ergänzen. Aus der Ablaufbeschreibung der Anwendungsfälle sind die Interaktionen zu bestimmen. Falls eine Beschreibung eines Ablaufs im Anwendungsfall zu ungenau ist, muss vorab der Fluss der Ereignisse textuell verfeinert werden. Für jeden internen Akteur als auch für die externen Akteure, wie zum Beispiel externe Systeme, sind Interaktionsklassen zum Klassenmodell hinzuzufügen. Weiterhin ist für jeden Anwendungsfall eine Vorgangsklasse, welche den Ablauf regelt, einzufügen. Zur Beschreibung des Interaktionsverhaltens sind UML-Sequenzdiagramme oder UML-Objektinteraktionsdiagramme (siehe [JBR99]) zu erstellen.

Bei der Analyse der Klassen sind für jede ermittelte Klasse des Analyseklassenmodells die Aufgaben der Klassen sowie Attribute, Assoziationen, Aggregationen und Generalisierungen festzulegen. Bei allen Aktivitäten der Analyse sind nichtfunktionale Anforderungen festzulegen.

8.1.3 Produktartefakte der Analyse

Zentrales Produkt der Analyse ist das Analysemodell. Im Analysemodell werden die Anforderungen an das System in der Sprache der Entwickler beschrieben. Die textuellen Ablaufbeschreibungen werden in Szenarios verfeinert, welche den Nachrichtenfluss zwischen Objekten beschreiben. Durch diese Transformation wird eine objektorientierte Sicht des Gesamtsystems erstellt. Das Analysemodell wird durch folgende Teilprodukte beschrieben:

- In einem UML-Klassendiagramm wird das *Klassenmodell der Analyse* dargestellt. Vorgangs-, Fach- und Interaktionsklassen [Bre01, JBR99] sind durch spezielle Stereotypen gekennzeichnet.

- *Anwendungsfallrealisierungen* werden in UML-Sequenzdiagrammen oder in UML-Objektinteraktionsdiagrammen (siehe [JBR99]) beschrieben.
- Die nichtfunktionalen Anforderungen, die über die gesamte Analysephase ermittelt werden, sind in einer textuellen Beschreibung der *Spezialanforderungen* zu ergänzen.
- Die *Analyse-Packages* werden mithilfe von UML-Package-Diagrammen [BRJ98] beschrieben.

Wie bereits erwähnt, muss gegebenenfalls die Ablaufbeschreibung zu einzelnen Anwendungsfällen ergänzt werden, falls die Szenarien aus den Abläufen nicht direkt hervorgehen. Änderungen an den Anwendungsfallbeschreibungen werden im Anwendungsfallmodell vorgenommen.

Abbildung 8.1: Szenario für den Anwendungsfall Korrekturbuchung

Abbildung 8.1 zeigt ein Szenario zur Realisierung des Anwendungsfalls *Korrekturbuchung* auf der *TimeTool*-Fallstudie. Der in Abbildung 7.1 beschriebene Ablauf wurde in einem exemplarischen Sequenzdiagramm umgesetzt. Zum Szenario wurden die Interaktionsklasse *Adjustment Posting UI* sowie die Vorgangsklassen *VG Adjustment Posting* und *Accounting Manager* ermittelt. Die Klasse *Accounting* ist eine Fachklasse und wurde bei der Transformation des Domänenmodells in das Klassenmodell der Analyse erstellt.

Im gezeigten UML-Sequenzdiagramm zum Szenario für den Anwendungsfall Korrekturbuchung sind auch die Stereotypen aus [JBR99, Bre01] für die Einteilung der Klassen in Fach-, Vorgangs- und Interaktionsklassen dargestellt.

8.2 Realisierung der Anwendungsfälle

In der Beschreibung der Aktivitäten zur Analyse zugriffssicherer Systeme in Abschnitt 8.1.2 haben wir bereits erwähnt, dass bei der *Realisierung der Anwendungsfälle* Vorgangs-, Fach- und Interaktionsklassen bestimmt und die exemplarischen Abläufe in Szenarien verfeinert werden.

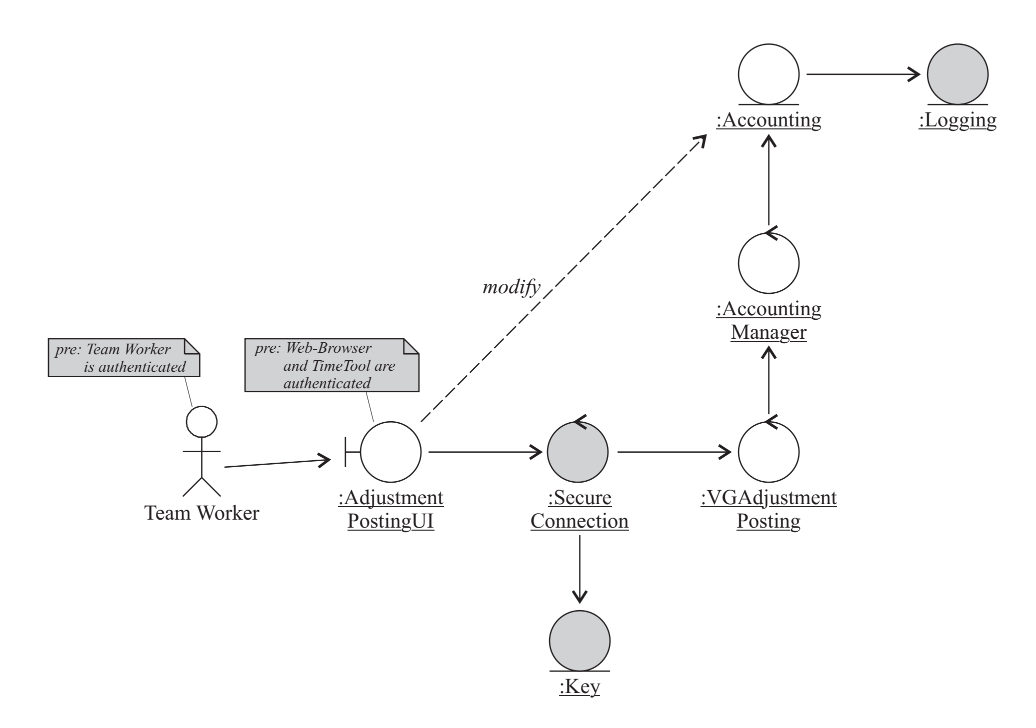


Abbildung 8.2: Szenario zur Korrekturbuchung mit Interaktion der Objekte

Bei der Spezifikation der Anwendungsfälle innerhalb der Anwendungsfallmodellierung zugriffssicherer Systeme haben wir die Funktionalität und Anforderungen an die Zugriffssicherheit spezifiziert. Zur Umsetzung der Aspekte der Zugriffssicherheit sind folgende Ergänzung und Verfeinerung der Szenarien in Abhängigkeit von den bereits ermittelten Schutzzielen durchzuführen:

Vertraulichkeit: Informationen sind durch einen eingeschränkten Benutzerzugriff auf Daten und Vorgängen sowie durch gesicherte Übertragungen vor unerlaubt lesendem Zugriff zu schützen. Komponenten zur sicheren Datenübertragung sind zum Analysemodell hinzuzufügen.

Integrität: Informationen sind durch einen eingeschränkten Benutzerzugriff auf Daten und Vorgängen sowie durch gesicherte Übertragungen vor unerlaubter Modifikation zu schützen. Komponenten zur sicheren Datenübertragung sind zum Analysemodell hinzuzufügen.

Verbindlichkeit: Änderungen an verbindlichen Informationen sind einer Beweissicherung zu unterziehen. Dabei sind neben den Modifikationen an Daten auch die ausführenden Akteure zu protokollieren. Komponenten und Vorgänge zur Beweissicherung sind im Analysemodell zu spezifizieren oder bei der Integration von Sicherheitsprotokollen (siehe Abschnitt 8.3) aufzunehmen.

Authentifikation: Zur Sicherstellung der Glaubwürdigkeit und Echtheit der Akteure und Objekte (beispielsweise Flussobjekte) sind Komponenten zur Durchführung der Authentifikation einzufügen. Diese Komponenten werden aus den Sicherheitsanwendungsfällen zur Authentifizierung abgeleitet.

Autorisierung: Die Benutzerrechte sind an geeigneten Stellen bei der Verfeinerung der Szenarien abzuprüfen. Methoden zur Überprüfung sind nach dem in Kapitel 4 vorgestellten Ansatz im Klassenmodell der Analyse zu integrieren.

Ebenso wie für die funktionalen Aspekte sind für Aspekte der Zugriffssicherheit geeignete Vorgangs-, Fach- und Interaktionsklassen in den Szenarien einzufügen und das Interaktionsverhalten ist zu verfeinern. Die verfeinerten Szenarien können in UML-Objektinteraktionsdiagrammen (vgl. [JBR99]) oder in UML-Sequenzdiagrammen dargestellt werden.

Abbildung 8.2 zeigt das Szenario zum Anwendungsfall Korrekturbuchung (siehe Abbildungen 7.1 und 7.8) als Objektinteraktionsdiagramm. Für die Anforderung an die Zugriffssicherheit *A1* aus der strukturellen Beschreibung des Sicherheitsanwendungsfalls in Abbildung 7.8 wurden Objekte der Vorgangsklasse *SecureConnection* und der Fachklasse *Key* in das Szenario aufgenommen. Zur Beweissicherung von Änderungen (Anforderung *A4)* an den *Accounting*-Einträgen wurde die Fachklasse *Logging* im Analyseklassendiagramm bestimmt, die bereits bei der Ermittlung der Bedrohungen, Risiken und Maßnahmen innerhalb der Geschäftsprozessmodellierung ermittelt wurde. In ihr werden vor der Änderung von Buchungsdaten die alten Buchungsdaten gespeichert. Durch die gerichteten Assoziationen wird der Informationsfluss abstrakt beschrieben.

Für die Anforderungen *A2* (Projektmitarbeiter muss authentifiziert sein) und *A3* (Web-Browser und TimeTool-System müssen gegenseitig authentifiziert sein) aus der Beschreibung des Sicherheitsanwendungsfalls sind eigene Szenarien zu entwerfen. Sie sind als Vorbedingung für das Szenario zur Korrekturbuchung auszuführen. Diese Vorbedingungen werden als Kommentare im Objektinteraktionsdiagramm und im Sequenzdiagramm in Abbildung 8.4 dargestellt. Alle Änderungen im Objektdiagramm, die sich aus der Verfeinerung der Sicherheitsaspekte aus dem Anwendungsfall Korrekturbuchung ergeben, sind in Abbildung 8.2 grau hinterlegt.

8.3 Integration von Sicherheitsprotokollen

Bei der Realisierung der Anwendungsfälle wurden zu den ermittelten Aspekten der Zugriffssicherheit die Techniken zur Realisierung ausgewählt. Hierbei wurde bereits festgelegt, welche Technik anzuwenden ist.

Authenticate With Chipcard

Team Worker
Authenticate ChipcardUI
:Chipcard
:Terminal
putCardIn
getChipNumber
CardNumber
askRandom
RandomNumber
encrypt(key, randomTerminal, randomChip)
encrypt(key, randomTerminal, randomChip)

Abbildung 8.3: Protokoll zur Authentifikation mittels Chipkarte und Terminal

Der genaue Ablauf wurde jedoch noch nicht beschrieben. Bei der weiteren Verfeinerung zur Realisierung der Anwendungsfälle sind nun die konkreten Protokolle und Techniken auszuwählen. Dabei sind zwei Fälle zu unterscheiden:

- Es existieren Protokolle, die zur Abdeckung der Zugriffssicherheit in das Analysemodell des zu entwickelnden Systems aufgenommen werden können. Gegebenenfalls sind Interaktions-, Vorgangs- und Fachklassen zum Klassenmodell der Analyse hinzuzunehmen. Der genaue Protokollablauf muss nicht spezifiziert werden, da im Design und in der Implementierung eine fertige Komponente eingesetzt wird. Hierbei kann es sich um eine Wiederverwendung aus einem bestehenden Softwareprodukt, um freien Code oder um eine Zukaufkomponente handeln. Für ein oder mehrere Protokolle können ein oder mehrere Analyse-Packages definiert werden.
- In vielen Fällen müssen Protokolle an das System angepasst werden oder es existieren keine Standardlösungen. In diesen Fällen ist der Protokollablauf in der Analyse zu modellieren.

In Abbildung 8.3 ist ein Spezialfall für eine Authentifikation exemplarisch modelliert worden. Der Zugang zum System wird hier nicht mittels Benutzername und Passwort überprüft, sondern durch die Verwendung einer Chipkarte und eines Terminals. Das exemplarisch spezifizierte Protokoll beschreibt eine gegenseitige symmetrische Authentisierung (siehe [RE99]) zwischen Chipkarte und Terminal.

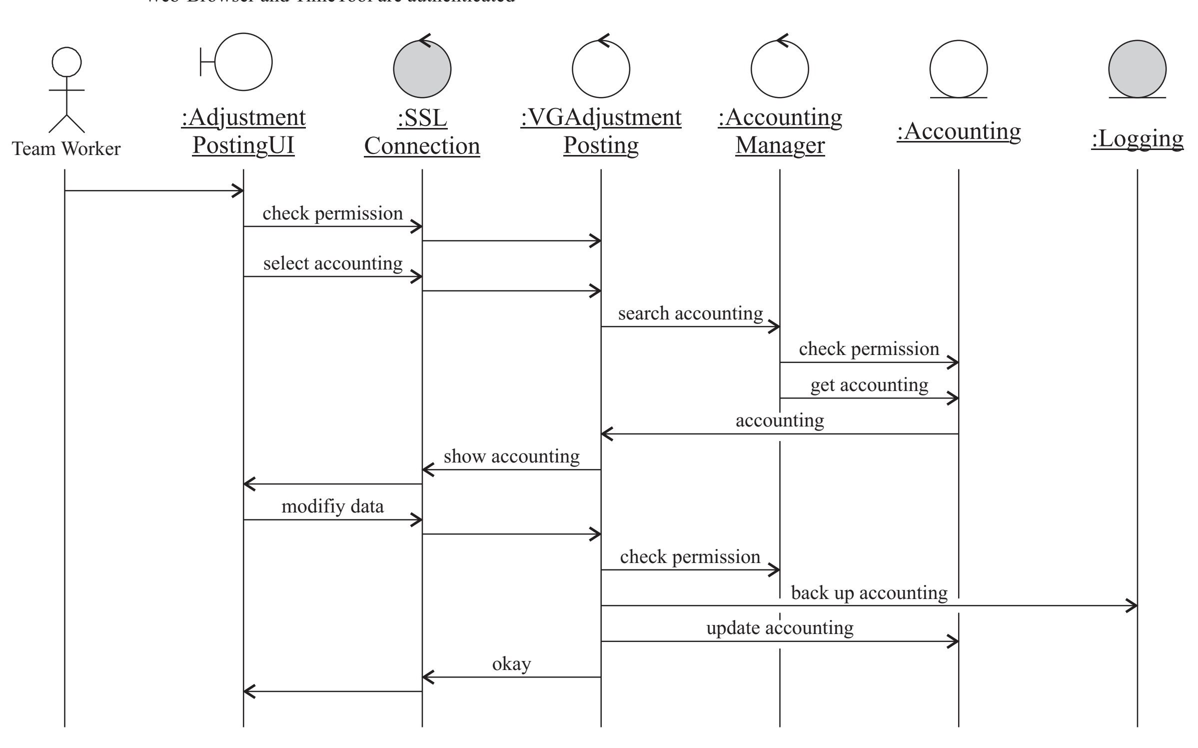


Abbildung 8.4: Verfeinertes Szenario zur Korrekturbuchung

In unserem bekannten Szenario zur Korrekturbuchung verwenden wir zur sicheren Datenübertragung der Benutzereingaben zwischen dem System des Benutzers (Web-Browser) und dem *TimeTool*-System das *Secure Socket Layer Protokoll* [FKK96, Oak01]. Hierzu wurde im Szenario eine Vorgangsklasse eingefügt, welche die Verschlüsselung, die sichere Übertragung und die Entschlüsselung übernimmt. Da wir dieses Protokoll im Design als fertige Komponente zum System hinzufügen, werden weitere Details und Subklassen zu diesem Protokoll nicht weiter spezifiziert; Einzelheiten zur Speicherung des Schlüssels, wie wir dies in Abbildung 8.2 bereits spezifiziert haben, werden nicht weiter betrachtet.

Für die Überprüfung der Autorisierung wurden Methodenaufrufe (*check permission*) eingefügt und der Ablauf der Beweissicherung an Buchungsobjekten wurde durch Interaktion mit der Klasse Logging spezifiziert. Die für die Abdeckung der Sicherheitsanforderungen hinzugefügten Objekte sind im Szenario zur Korrekturbuchung in Abbildung 8.4 grau hinterlegt.

8.4 Der Prozess der Analyse zugriffssicherer Systeme

In diesem Kapitel haben wir bereits die Aktivitäten der Analyse zugriffssicherer Systeme zur Realisierung der Anwendungsfälle und zur Integration von Sicherheitsprotokollen vorgestellt. Im Folgenden zeigen wir die Eingliederung dieser Aktivitäten in den Entwicklungsprozess (Abschnitt 8.4.1) sowie die erneute Anwendung des Sicherheitsmikroprozesses (Abschnitt 8.4.2).

8.4.1 Prozessmuster 8.1: Analyse zugriffssicherer Systeme

Kurzbeschreibung

Kernziele der Analyse sind die Transformation des Domänenmodells in ein Klassendiagramm der Analyse mit Fach-, Vorgangs- und Interaktionsklassen sowie die Realisierung der Anwendungsfälle in Vorgangsklassen. Für eine durchgängige und sukzessive Entwicklung der Sicherheit sind die bereits in Struktur und Verhalten spezifizierten Aspekte der Zugriffssicherheit in die neuen Modelle zu transformieren und zu verfeinern.

Ergänzungen zur Gewährleistung der Zugriffssicherheit in Struktur und Funktionalität sind erneut nach Aspekten der Zugriffssicherheit zu untersuchen.

Problem

Bei der Analyse zugriffssicherer Systeme werden aus den Anwendungsfällen und Klassen des Domänenmodells Fach-, Vorgangs- und Interaktionsklassen abgeleitet. Bekannte Verfahren zur Analyse von Systemen, wie zum Beispiel [Kru00] und [JBR99], konzentrieren sich auf die Entwicklung der Funktionalität und betrachten die Sicherheitsaspekte nur im Rahmen der Ermittlung von nichtfunktionalen Anforderungen. Ein strukturiertes Vorgehen dazu wird nicht beschrieben.

Die in den Anwendungsfällen und im Domänenmodell gewonnenen Aspekte der Zugriffssicherheit sind jedoch in die Modelle der Analyse zu überführen. Dabei müssen die textuell beschriebenen Anforderungen an das zu entwickelnde System in präzisere Modelle in der Sprache der Entwickler verfeinert werden.

Im Rahmen der Analyse müssen im Klassendiagramm der Analyse zur Lösung der Anforderungen an die Zugriffssicherheit neue Klassen aufgenommen werden. Da diese neuen Klassen noch keiner Analyse zugriffssicherer Systeme unterzogen wurden, sind hier potenzielle Angriffe möglich. Diese Ergänzungen sind bei der Analyse zugriffssicherer Systeme zu betrachten.

Da die Beschreibungen der Struktur und des Verhaltens geändert werden, wird die Beschreibung der Benutzerrechte inkonsistent. Die Benutzerrechtespezifikation ist anzupassen.

Lösung

Für die durchgängige und sukzessive Modellierung von Aspekten der Zugriffssicherheit sind die erarbeiteten Sicherheitsanforderungen aus dem Domänenmodell und aus dem Anwendungsfallmodell in die verfeinerte Struktur aus Fach-, Vorgangs- und Interaktionsklassen zu transformieren und zu verfeinern.



Abbildung 8.5: Die Aktivitäten der Analyse zugriffssicherer Systeme

Bei der Verfeinerung der Anwendungsfälle sind die Anforderungen an die Zugriffssicherheit in die zu entwickelnden Szenarien zu integrieren. Für die spezifizierten Schutzziele sind geeignete Anpassungen und Ergänzungen im Funktionsablauf und im Klassendiagramm durchzuführen, so sind beispielsweise für eine sichere Übertragung Klassen zur Verschlüsselung einzuführen und Methoden zur Verschlüsselung der Daten im Ablauf aufzurufen. Für die Realisierung der Anwendungsfälle sind Vorgangsklassen im Klassenmodell der Analyse mit aufzunehmen, die später den eigentlichen Ablauf in *process*-Methoden beinhalten.

Die in den Klassen des Domänenmodells spezifizierten Schutzziele wurden bereits innerhalb der Geschäftsprozessmodellierung zugriffssicherer Systeme als auch bei der Anwendungsfallmodellierung zugriffssicherer Systeme zur Ermittlung von Bedrohungen, Maßnahmen sowie zur Benutzerrechtemodellierung verwendet. Diese Schutzziele sind bereits durch die Gegenmaßnahmen abgedeckt und müssen nicht in die Klassen des Analysemodells übertragen werden.

Für Klassen, die im Klassendiagramm des Analysemodells während der Analysephase hinzugefügt werden, insbesondere für die neu hinzugefügten Vorgangsklassen, sind die Bedrohungen, Risiken und Maßnahmen durch die wiederholte Ausführung des Sicherheitsmikroprozesses aus Abschnitt 5.3.3 zu überprüfen. Die ermittelten Maßnahmen sind in der Analysephase umzusetzen.

Die Benutzerrechte sind auf das Analysemodell abzubilden, insbesondere sind die Ausführungsrechte von Anwendungsfällen auf die *process*-Methoden der Vorgangsklassen zu überführen. Diejenigen Benutzerrechte, die noch textuell beschrieben sind, müssen prädikativ spezifiziert werden.

Abbildung 8.5 gibt einen Überblick über die Aktivitäten der Analyse zugriffssicherer Systeme. Da in allen Aktivitäten Aspekte der Zugriffssicherheit betrachtet werden, sind alle Aktivitäten grau hinterlegt.

Aktivitäten

Innerhalb der Analyse zugriffssicherer Systeme sind bei der Prozessausführung die folgenden Aktivitäten nach der in der Lösung und in der Struktur skizzierten Vorgehensweise auszuführen. Aus Gründen der Vollständigkeit wurden bei den Aktivitäten auch diejenigen aus der allgemeinen Analyse mit aufgenommen und umrissen (vgl. Abschnitt 8.1.2).

- *Identifikation der Analyse-Packages*
  Analysepakete dienen zur Aufteilung des Analysemodells in kleinere und handhabbarere Einheiten. Um die Analyse schrittweise durchführen zu können, sind die *Packages* zu Beginn zu ermitteln. Zur Umsetzung von Anwendungsfällen zu Sicherheitsgrundfunktionen können eigene Analyse-Packages definiert werden. Ein Analysepaket enthält eine Menge von Anwendungsfallrealisierungen und die dazugehörige Menge von Klassen des Analyseklassendiagramms.

- *Identifikation der Fach- und Interaktionsklassen*
  Das Domänenmodell ist in das Klassendiagramm der Analyse zu transformieren und dabei sind die Klassen in Fach- und Interaktionsklassen einzuteilen. Klassen des Anwendungskerns sowie die bereits spezifizierten Ein- und Ausgabetypen von Anwendungsfällen werden zu Fachklassen, die Schnittstellen zu externen Systemen hingegen zu Interaktionsklassen. Klassen, die zur Abwehr von Bedrohungen zum Domänenmodell hinzugefügt wurden, sind ebenfalls im Klassenmodell der Analyse zu verfeinern, so zum Beispiel die Klasse *Logging* aus der *TimeTool*-Fallstudie.

- *Realisierung der Anwendungsfälle*
  Die Anwendungsfälle aus der Anwendungsfallmodellierung sind in so genannten *Szenarien* zu verfeinern. Dazu sind die interagierenden Klassen und die Interaktionen zwischen Objekten der Klassen zu spezifizieren. Für jeden Anwendungsfall ist eine Vorgangsklasse zu erstellen, in der später der Ablauf des Anwendungsfalls in einer *process*-Methode implementiert wird. Alle Vorgangsklassen sowie an den Interaktionen beteiligte Klassen, die noch kein Bestandteil des Klassendiagramms der Analyse sind, müssen dem Klassendiagramm hinzugefügt werden. Aspekte der Zugriffssicherheit, die in den Anwendungsfällen textuell spezifiziert wurden, sind in der Ablaufspezifikation, d.h., in den Szenarien, zu integrieren. Notwendige Klassen zur Realisierung der Sicherheitsfunktionen sind ebenfalls dem Klassendiagramm hinzufügen. Auf die Aspekte der Zugriffssicherheit bei der Realisierung der Anwendungsfälle wurde im Abschnitt 8.2 eingegangen.

- *Protokollauswahl und -modellierung*
  Zur Gewährleistung der Funktionalität der Sicherheit sind geeignete Protokolle auszuwählen oder zu entwickeln. Diese Protokollabläufe oder der Aufruf von Protokollen sowie die notwendigen Klassen sind den Szenarien und dem Klassendiagramm der Analyse hinzufügen. Die Identifikation von Sicherheitsprotokollen wurde in Abschnitt 8.3 diskutiert.

- *Analyse der Spezialanforderungen*
  Die nichtfunktionale Anforderung der Zugriffssicherheit wurde in der Anforderungsanalyse zugriffssicherer Systeme detailliert behandelt. Es gibt jedoch eine Reihe weiterer nichtfunktionaler Anforderungen, die bei der Systementwicklung betrachtet werden

müssen. Innerhalb der Analyse sind diese Anforderungen zu ermitteln und zu integrieren (siehe [JBR99]). Zur Analyse der so genannten Spezialanforderungen sind die Beschreibungen zu den Ergänzungen aus dem Geschäftsprozessmodell sowie zur Anforderungsergänzung aus dem Anwendungsfallmodell heranzuziehen.

- *Analyse der Klassen*
  Für alle Klassen des Klassendiagramms der Analyse sind die Beziehungen und Attribute zu verfeinern bzw. zu ergänzen. Basierend auf den entwickelten Szenarien sind die Verantwortlichkeiten der Klassen festzulegen.

- *Ausführung des Sicherheitsmikroprozesses*
  Bei der Realisierung der Anwendungsfälle werden weitere Klassen zum Klassendiagramm der Analyse hinzugefügt. Da diese Klassen noch keiner Sicherheitsanalyse unterzogen wurden, muss im Rahmen der Analyse für diese Klassen der Sicherheitsmikroprozess aus Abschnitt 5.3.3 erneut ausgeführt werden. Die ermittelten Maßnahmen gegen Bedrohungen mit nicht vermeidbarem Risiko sind innerhalb der Analyse zugriffssicherer Systeme in den Szenarien zu integrieren.

- *Modellierung der Ausführungsrechte*
  Innerhalb der Anwendungsfallmodellierung wurden die Ausführungsrechte für Anwendungsfälle spezifiziert. Die Anwendungsfälle wurden auf Szenarien abgebildet, die in der Designphase auf die bereits im Klassendiagramm integrierten Vorgangsmethoden transformiert werden. Im Abschnitt 6.3.3 haben wir bereits *process*-Methoden für die Spezifikation der Abläufe in Vorgangsklassen vorgestellt. Die zu den Anwendungsfällen spezifizierten Benutzerrechte sind nun in Benutzerrechte für die *process*-Methoden zu transformieren. Da jedoch die Anwendungsfälle direkt auf die Vorgangsklassen abgebildet werden, d.h., zu jedem Anwendungsfall wird eine Vorgangsklasse entwickelt, können die Ausführungsrechte direkt übertragen werden.

  Im Rahmen der Geschäftsprozessmodellierung und der Anwendungsfallmodellierung zugriffssicherer Systeme wurden schon alle Ausführungsrechte, die nicht mehr abgeändert werden mussten, prädikativ spezifiziert. Für alle Ausführungsrechte, die zu Beginn der Analyse zugriffssicherer Systeme noch in textueller Form vorliegen, sind die Ausführungsrechte endgültig zu klären und in eine prädikative Form zu überführen, sodass die Überprüfung der Benutzerrechte (Autorisierung) im Rahmen des folgenden Designs automatisch generiert werden kann (siehe Abschnitte 4.4 und 6.3.3).

- *Modellierung der Zugriffsrechte*
  Die Klassen des Domänenmodells werden auf Klassen des Klassendiagramms der Analyse abgebildet. Da die Klassen dabei nur neu eingeteilt werden, die Struktur und Assoziationen aber erhalten bleiben, können die Zugriffsrechte aus der Anwendungsfallmodellierung beibehalten werden. Durch die externe Beschreibung der Zugriffsrechte in der Benutzerrechtematrix sind keine Änderungen an der Matrix erforderlich.

  Werden innerhalb der Analyse zugriffssicherer Systeme noch weitere Fach- oder Interaktionsklassen zum Klassendiagramm hinzugefügt, so sind gegebenenfalls weitere Zugriffsrechte zu spezifizieren. Diese leiten sich aus der Anwendung des Sicherheitsmikroprozesses ab.

  Ebenso wie bei der Modellierung der Ausführungsrechte sind alle noch textuell spezifizierten Zugriffsrechte in eine prädikative Form zu verfeinern.

Produktartefakte

Die folgende Auflistung beschreibt die Eingaben und Ausgaben zum Prozess der Analyse zugriffssicherer Systeme. Um Sicherheitsaspekte erweiterte Standardproduktartefakte sind gekennzeichnet.

Eingabe-Produktartefakte

- Business View Modell (aus der Startphase)
- Domänenmodell (mit Sicherheitsaspekten erweitert)
- Geschäftsprozessmodell (mit Sicherheitsaspekten erweitert)
- Anwendungsfallmodell (mit Sicherheitsaspekten erweitert)
- Bedrohungs- und Risikomodell
- Benutzerrechtemodell

Ausgabe-Produktartefakte

- Anwendungsfallmodell (mit Sicherheitsaspekten erweitert)
- Analysemodell (mit Sicherheitsaspekten erweitert)
- Bedrohungs- und Risikomodell (erweitert und verfeinert)
- Benutzerrechtemodell (erweitert und verfeinert)

Kontext

Dieses Prozessmuster kann nur im Rahmen einer Anforderungsspezifikation zugriffssicherer Systeme (vgl. Abschnitt 5.3.2), basierend auf einem objektorientierten Vorgehensmodell, ausgeführt werden. Für die Ausführung des Prozessmusters werden mit Schutzzielen annotierte Klassen und mit Sicherheitsaspekten ergänzte Anwendungsfälle vorausgesetzt. Weiterhin müssen die Benutzerrechte in einer Benutzerrechtematrix spezifiziert werden. Der Fokus dieser Analyse zugriffssicherer Systeme liegt auf einer Neuentwicklung eines betrieblichen Informationssystems. Aktivitäten zur Analyse bestehender Systeme sind kein Bestandteil dieses Prozessmusters.

Neben den Entwicklungsschritten sind auch projektübergreifend Managementaufgaben durchzuführen, auf die jedoch im Rahmen dieses Prozessmusters nicht näher eingegangen wird.

Struktur

Innerhalb der Lösung wurde bereits der Ablauf der Analyse zugriffssicherer Systeme skizziert. Im Aktivitätsdiagramm in Abbildung 8.6 ist der verfeinerte Ablauf dargestellt. Zudem werden auch geeignete Iterationsmöglichkeiten aufgezeigt.

Die Entwicklungsphase der Analyse kann inkrementell ausgeführt werden. Insbesondere können die Analyse-Packages, die zu Beginn der Analyse zugriffssicherer Systeme definiert werden, schrittweise umgesetzt werden. Die Identifikation von Klassen und die Realisierung der Anwendungsfälle kann vollständig oder nur teilweise abgeschlossen sein, bevor einzelne Klassen im Detail spezifiziert werden. Nach der Analyse der Klassen steht es dem Entwickler offen, weitere Packages oder Klassen und Anwendungsfallrealisierungen zu erarbeiten oder Bedrohungen und Risiken durch die Ausführung des Sicherheitsmikroprozesses zu ermitteln. Ebenso können Berechtigungen jeweils nach Ermittlung der Schutzziele oder am Ende der Analyse zugriffssicherer Systeme spezifiziert werden.

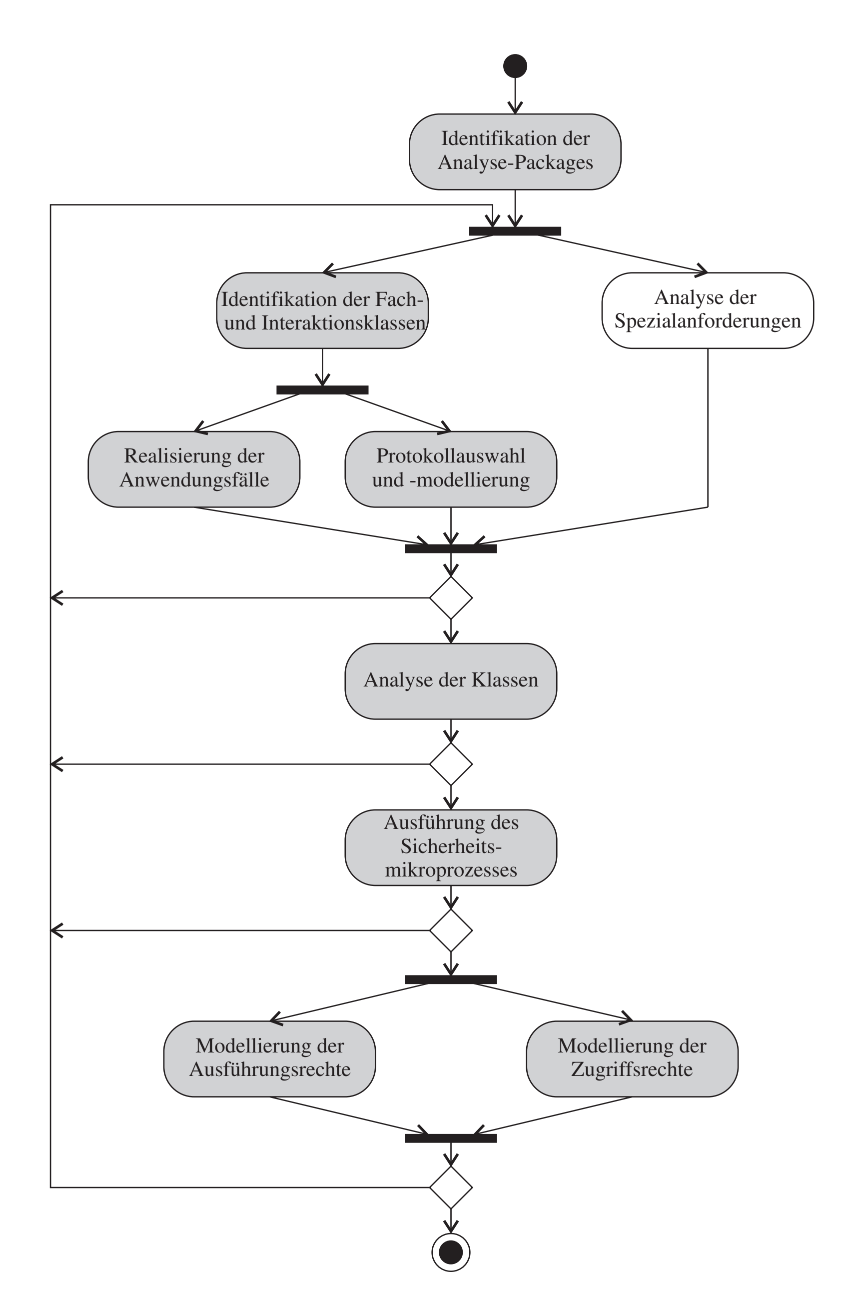


Abbildung 8.6: Der Ablauf der Analyse zugriffssicherer Systeme

Durch den vorgestellten Prozess wird die gemeinsame Entwicklung von Funktionalität und Sicherheit gefordert. Sobald Aspekte der Zugriffssicherheit betrachtet werden können, schreibt der Prozess die Behandlung von Sicherheitsaspekten vor. Schutzziele, Bedrohungen, Maßnahmen und Benutzerrechte müssen vor Abschluss der Analyse zugriffssicherer Systeme ermittelt werden.

Vor- und Nachteile

Durch das vorgestellte Prozessmuster wird die frühzeitige Entwicklung von Sicherheit innerhalb der Phase der Anforderungsdefinition weiter unterstützt. Durch die Betrachtung von

Sicherheitsaspekten in allen Prozessabschnitten der Anforderungsspezifikation zugriffssicherer Systeme wird eine durchgehende Sicherheitsanalyse gefordert. Insbesondere wird durch den vorgestellten Prozess die geforderte sukzessive Entwicklung erreicht, da die Ergebnisse aus der Anwendungsfallmodellierung zugriffssicherer Systeme in die Analyse zugriffssicherer Systeme einfliesen und dabei verfeinert und ergänzt werden.

Wie bereits im Domänenmodell bleibt das Klassendiagramm der Analyse frei von Benutzerrechten. Hierzu werden Zugriffs- und Ausführungsrechte im Benutzerrechtemodell in einer Benutzerrechtematrix spezifiziert. In dieser Benutzerrechtematrix können sowohl Zugriffs- als auch Ausführungsrechte einheitlich dargestellt werden.

Das Prozessmuster zur Analyse zugriffssicherer Systeme bietet, wie bereits erwähnt, eine Vielzahl von Iterationsmöglichkeiten (vgl. Struktur des Prozessmusters). Besonders für große Systeme spielt die iterative Entwicklung eine bedeutende Rolle. Eine entscheidende Rolle spielt dabei die Tatsache, dass die Entwicklung der Sicherheitsaspekte nicht in den Hintergrund treten darf, d.h., dass Sicherheitsaspekte in einer eigenen Iteration am Ende isoliert ermittelt werden. Bei den vorgestellten Iterationsmöglichkeiten kann das System in kleineren Inkrementen entwickelt werden, Sicherheitsaspekte sind dabei in jeder Iteration durchgehend zu ermitteln, da diese an die funktionale Entwicklung gebunden sind.

Wie bereits bei den vorausgehenden beiden Schritten der Anforderungsspezifikation zugriffssicherer Systeme erwähnt wurde, erfordert dieses Verfahren durch die integrierte Entwicklung der Sicherheitsaspekte zusätzlichen Aufwand. Schutzziele, Bedrohungen, Risiken, Maßnahmen und Berechtigungen sind in jeder Phase erneut zu überprüfen und gegebenenfalls zu ergänzen. Wie bereits erwähnt, reicht es bei zugriffssicheren Systemen nicht aus, die Sicherheitsanforderungen wie funktionale Anforderungen einmalig zu erfassen, da alle potenziellen Angriffe und somit Verletzungsmöglichkeiten des Systems erfasst werden müssen.

In Beziehung stehende Prozessmuster

Übergeordneter Prozess

- Prozessmuster zur Anforderungsdefinition zugriffssicherer Systeme (Abschnitt 5.3.2)

Auszuführender Teilprozess

- Prozessmuster zum Sicherheitsmikroprozess (siehe Abschnitt 5.3.3)

Weitere referenzierte Prozessmuster

- Prozessmuster zur Geschäftsprozessmodellierung zugriffssicherer Systeme (siehe Abschnitt 6.4.1)
- Prozessmuster zur Anwendungsfallmodellierung zugriffssicherer Systeme (vergleiche Abschnitt 7.5.1) ◇

8.4.2 Anwendung des Sicherheitsmikroprozesses

Wie in den beiden vorausgehenden Phasen der Anforderungsspezifikation zugriffssicherer Systeme kommt auch bei der Analyse zugriffssicherer Systeme der Sicherheitsmikroprozess aus Abschnitt 5.3.3 erneut zur Anwendung. Dabei werden Bedrohungen, Risiken und Maßnahmen zu folgenden Änderungen und Ergänzungen ermittelt:

- Zur Realisierung der Sicherheitsfunktionen sind im Klassendiagramm Vorgangsklassen und Fachklassen zu ergänzen. Die Vorgangsklassen enthalten nach der Entwicklung des Designs die Sicherheitsgrundfunktionen, die zur Gewährleistung der Zugriffssicherheit eingesetzt werden. Dabei sind gegebenenfalls Attribute vor unerlaubtem Zugriff zu schützen. In den Fachklassen zur Sicherheit werden zusätzliche Informationen gehalten, wie beispielsweise private und allgemeine Schlüssel, die ebenfalls vor unerlaubtem Zugriff zu schützen sind.

- Bei der Spezifikation der Szenarien werden zum Klassendiagramm Vorgangsklassen ermittelt und wenn nötig Interaktions- und Fachklassen ergänzt.

- Zu den hinzugefügten Klassen sind im Klassenmodell neue Beziehungen einzufügen.

Da die Analyse zugriffssicherer Systeme der letzte Prozessabschnitt der dreistufigen Anforderungsspezifikation ist, müssen die ermittelten Maßnahmen noch in diesem Abschnitt umgesetzt werden. Im Gegensatz zu den vorausgehenden Prozessabschnitten ist es nicht möglich, die Maßnahmen erst im nächsten Abschnitt zu integrieren.

8.5 Produktartefakte der Analyse zugriffssicherer Systeme

Abschließend betrachten wir die Produktartefakte aus dem Prozessmuster zur Analyse zugriffssicherer Systeme und gehen dabei auf die notwendigen Ergänzungen zur Modellierung der Zugriffssicherheit näher ein.

In Abbildung 8.7 sind die Produktartefakte zur Sicherheitsanalyse mit ihren Teilprodukten dargestellt. Dabei sind alle mit Sicherheitsaspekten veränderten oder hinzugefügten Teilprodukte grau hinterlegt. Veränderungen an den Produkten des Produktmodells sind in der in Abschnitt 6.5 vorgestellten Art gekennzeichnet.

Die Elemente des *Business View Modells* sowie des *Geschäftsprozessmodells* werden bei der Analyse verwendet und bleiben unverändert.

Falls bei der Entwicklung der Szenarien festgestellt wird, dass die Beschreibungen der Anwendungsfälle für die Umsetzung zu unpräzise sind, müssen die Beschreibungen angepasst werden. Der verfeinerte Fluss der Ereignisse wird dann in der Anwendungsfallbeschreibung im *Anwendungsfallmodell* verfeinert. Die weiteren Teilprodukte der Anwendungsfallmodellierung werden verwendet und bleiben ebenfalls unverändert.

Erstmals innerhalb der Anforderungsspezifikation zugriffssicherer Systeme werden Änderungen am Klassenmodell nicht mehr im *Domänenmodell* vorgenommen, sondern im Klassenmodell der Analyse. Neben diesem Teilprodukt besteht das neu hinzugefügte *Analysemodell* aus den Anwendungsfallrealisierungen, der Beschreibung der Spezialanforderungen und der Analyse-Packages.

Die Zugriffs- und Ausführungsrechte werden im *Benutzerrechtemodell* angepasst und Bedrohungen, Risiken, Maßnahmen und Überprüfungsergebnisse werden im *Bedrohungs- und Risikomodell* ergänzt oder modifiziert.

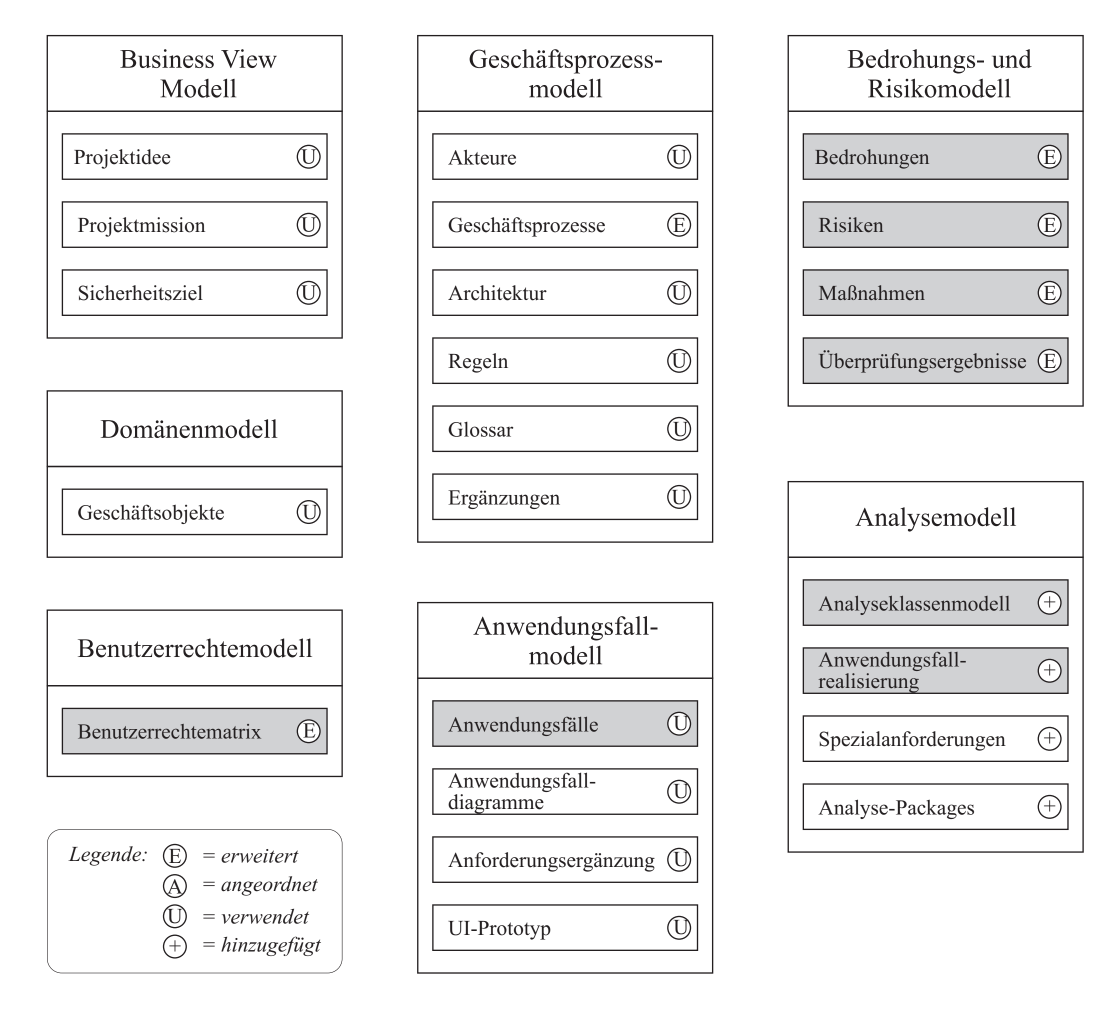


Abbildung 8.7: Produktartefakte der Analyse zugriffssicherer Systeme

8.6 Zusammenfassung

Ziel der Analyse zugriffssicherer Systeme ist es, die durchgängige und sukzessive Entwicklung von Aspekten der Zugriffssicherheit fortzusetzen und in funktionale Anforderungen zu überführen. Hierzu werden die Struktur der Anwendung in das Klassendiagramm der Analyse verfeinert, im Verhalten die Sicherheitsfunktionalität in Szenarien integriert und die Benutzerrechte verfeinert.

Innerhalb der Analyse sind die Klassen aus dem Domänenmodell der Anwendungsfallmodellierung zur Sicherheit in das Klassendiagramm der Analyse zu transformieren. Die Schutzziele mit ihren Bedrohungen, Risiken und Maßnahmen müssen hier nicht erneut in die Klassen aufgenommen werden, da zur Behandlung der Bedrohungen mit nicht geringfügigem Risiko bereits Maßnahmen ermittelt und in die funktionalen Anforderungen integriert worden sind. Die Benutzerrechte aus dem Domänenmodell werden für die Klassen des Analysemodells übernommen.

Die Anforderungen an das Verhalten werden innerhalb der Analyse zugriffssicherer Systeme von textuell spezifizierten Anwendungsfällen in semiformale Ablaufszenarien verfeinert. Dabei

sind die Anforderungen an die Zugriffssicherheit, die innerhalb der Anwendungsfälle strukturell beschrieben wurden, in funktionale Abläufe und Klassen zu verfeinern. Für die Änderungen am Klassendiagramm der Analyse und die hinzugefügten Vorgangs-, Interaktions- und Fachklassen müssen die Benutzerrechte transformiert oder neu ermittelt werden.

Für eine automatische Codegenierung zur Überprüfung der Benutzerrechte müssen diese am Ende der Analyse zugriffssicherer Systeme in einer prädikativen Form vorliegen. Alle Benutzerrechte, die vorab nur informal spezifiziert worden sind, sind innerhalb der Analyse zu formalisieren.

Neben der durchgängigen, sukzessiven und gemeinsamen Entwicklung von Funktionalität und Sicherheit wurde auch das Ziel der lokalen Ermittlung der Sicherheit erfüllt. Bei der Realisierung der Anwendungsfälle werden die Aspekte der Zugriffssicherheit in kleinen Einheiten umgesetzt.

Die iterative Entwicklung ist durch die Anwendung des Sicherheitsmikroprozesses sowie durch weitere Iterationsmöglichkeiten innerhalb des Prozessablaufs der Analysemodellierung gegeben.

9 Zusammenfassung und Ausblick

In zugriffssicheren Systemen sind die dem System anvertrauten Informationen vor unautorisiertem Zugriff zu schützen. Heutige Vorgehensmodelle widmen sich nur in geringem Umfang der systematischen Entwicklung von Aspekten der Zugriffssicherheit, sodass diese häufig erst am Ende der Designphase in einem zusätzlichen Entwicklungsschritt ermittelt werden. Da in diesem Fall zur Abdeckung der Sicherheitsanforderungen große Teile des Designs grundlegend zu überarbeiten sind und zumeist die Projekte am Ende der Entwicklung die Zeit- und Kostenkalkulationen überschritten haben, bleiben viele Anforderungen an die Zugriffssicherheit unerfüllt. Dazu haben die "sichtbaren Funktionen" stets Entwicklungspriorität, nichtfunktionale Eigenschaften werden oftmals nur in untergeordnetem Maße behandelt. Dies führt dazu, dass die zu entwickelnden Systeme mit Sicherheitslücken und Schwachstellen behaftet sind.

Die vorgestellte Methode zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Informationssysteme beugt die Entwicklung lückenhafter Systeme vor, indem sie die Aspekte der Zugriffssicherheit in die frühen Phasen der Systementwicklung sinnvoll und leicht anwendbar integriert. Auf Basis eines objektorientierten System- und Vorgehensmodells beschreibt sie einen Ausschnitt zur durchgängigen und sukzessiven Neuentwicklung von zugriffssicheren Informationssystemen. Durch die vorgestellte Methode wird dem Anforderungsentwickler ein Instrumentarium an die Hand gegeben, sodass er die Anforderungen an die Zugriffssicherheit von technischen Details abstrahierend, gemeinsam mit der Entwicklung der funktionalen Anforderungen an das System schrittweise erarbeiten kann.

Zentrales Element für die Sicherheitsanalyse ist ein Benutzerrechtemodell zur Spezifikation von Zugriffs- und Ausführungsrechten in den frühen Phasen der Entwicklung. Diese basiert auf der Sprache P-MOS zur Navigation über Objekte, Zustände und Objektbezüge, sodass Benutzerrechte im Rahmen eines objektorientierten Vorgehens formalisiert werden können. Im Gegensatz zu existierenden Ansätzen werden Benutzerrechte außerhalb des Anwendungskerns innerhalb einer Berechtigungsmatrix vorab textuell spezifiziert und im Rahmen einer dreistufigen Sicherheitsanalyse in Hinblick auf eine spätere Generierung der Methoden zur Berechtigungsüberprüfung formalisiert. Für eine Integration der Berechtigungsüberprüfung wird die Methodenspezifikationen um einen Berechtigungsabschnitt erweitert, in dem die Zugriffsrechte beim Aufruf einer Objektmethode vorab überprüft werden können.

Existierende Methoden zur Entwicklung von sicheren Systemen wurden vorab zur Entwicklung von Sicherheitsanforderungen untersucht. Basierend auf den Ergebnissen wurden Anforderungen an einen Prozess zur Entwicklung sicherer Systeme abgeleitet. Da sich die Arbeit auf den Ausschnitt der Anforderungsdefinition zugriffssicherer Systeme beschränkt, wird die Eingliederung der Sicherheitsanalyse in den Softwarelebenszyklus gezeigt. Zur Ermittlung der Sicherheitsanforderungen wurde der dreistufige Prozess der Anforderungsspezifikation zugriffssicherer Systeme entwickelt. Zur Ermittlung von Sicherheitsanforderungen in den einzelnen Abschnitten der Anforderungsspezifikation zugriffssicherer Systeme kommt ein iterativ

ausführbarer Sicherheitsmikroprozess zum Einsatz. Im Hinblick auf eine Integration der vorgestellten Prozesse in verschiedene objektorientierte Vorgehensmodelle wurden die Prozesse mithilfe von Prozessmustern beschrieben.

Die drei Abschnitte der Sicherheitsanalyse, die Geschäftsprozessmodellierung, die Anwendungsfallmodellierung und die Analyse zugriffssicherer Systeme, werden auf eine durchgehende und sukzessive Modellierung von Aspekten der Zugriffssicherheit untersucht. In allen Abschnitten werden die Produktartefakte ermittelt, die zur Spezifikation der Sicherheitsaspekte zu verwenden, zu erweitern und zu ergänzen sind. Bei der Beschreibung der Prozesse werden die Aktivitäten zur Ermittlung der Sicherheitsaspekte und die Integration der Entwicklung von Sicherheitsanforderungen in die funktionalen Anforderungen gezeigt. Die Iterationsmöglichkeiten der Prozesse, insbesondere auch die iterative Anwendung des Sicherheitsmikroprozesses zur Ermittlung von Bedrohungen, Risiken und Maßnahmen sowie zur Überprüfung der Maßnahmen, werden dargestellt. Weiterhin wird die Spezifikation der Benutzerrechte auf Basis des vorgestellten Benutzerrechtemodells in den verschiedenen Abschnitten der Anforderungsspezifikation zugriffssicherer Systeme durchgeführt. Neben diesen allgemeinen Aspekten werden in den Prozessabschnitten der Anforderungsspezifikation spezifische Aktivitäten zur Modellierung der Zugriffssicherheit ausgeführt.

Bei der Geschäftsprozessmodellierung zur Sicherheit steht die Spezifikation von Schutzzielen in Struktur und Verhalten im Vordergrund, die gemeinsam mit Anwendern, Auftraggebern, Auftragnehmern und Anforderungsentwicklern erarbeitet wird. Die Bedeutung der Schutzziele im Kontext der Struktur- und Verhaltensbeschreibung wird geklärt und mithilfe des Erweiterungsmechanismus der UML werden Stereotypen für Schutzziele eingeführt. Bei der Modellierung der Benutzerrechte werden eine Hierarchie von Akteuren erstellt und verschiedene Typen von Ausführungsrechten definiert.

Bei der Anwendungsfallmodellierung zur Sicherheit steht die Modellierung des Schutzziels Authentifikation im Vordergrund. Für UML-Aktivitätsdiagramme wird das Stereotyp Authentifikation eingefügt und das Sitzungskonzept integriert. Die Abläufe mit annotierten Aktivitäten zur Authentifikation werden in Aktivitäten zur Überprüfung der Authentifikation und der Autorisierung verfeinert. Für die Verhaltensbeschreibung der Anforderungen an die Zugriffssicherheit werden die strukturellen Anwendungsfallbeschreibungen ergänzt. Für die Realisierung der Sicherheitsgrundfunktionen werden der funktionalen Anforderungsspezifikation eigene Sicherheitsanwendungsfälle hinzugefügt. Benutzerrechte, die bereits nach der Anwendungsfallmodellierung stabil sind, werden von der textuellen in eine prädikative Form präzisiert. Der Zusammenhang zwischen Ausführungs- und Zugriffsrechten wird während der Ablaufmodellierung berechnet. Die Grenzen der Berechenbarkeit sowie die Berechnungsvorschrift werden im Rahmen der Anwendungsfallmodellierung zugriffssicherer Systeme vorgestellt.

Innerhalb der Sicherheitsanalyse wird das Domänenmodell durch ein Klassendiagramm der Analyse mit Fach-, Vorgangs- und Interaktionsklassen verfeinert. Aus den Anwendungsfallbeschreibungen werden Szenarien mit exemplarischen Interaktionsverhalten erstellt. Die Sicherheitsanforderungen werden dabei ebenfalls wie die funktionalen Anforderungen durch funktionale Abläufe verfeinert. An dieser Stelle erfolgt der Übergang der nichtfunktionalen Sicherheitsanforderungen in funktionale Lösungen zur Gewährung der Zugriffssicherheit. Dabei sind zur Realisierung der Sicherheitsanforderungen geeignete Protokolle auszuwählen oder zu konstruieren. Innerhalb der Sicherheitsanalyse kommt auch der Sicherheitsmikroprozess zur

erneuten Anwendung. Werden hier neue Bedrohungen mit nicht minderem Risiko ermittelt, so sind die ermittelten Maßnahmen noch innerhalb dieses Prozessabschnitts umzusetzen, da die dreistufige Anforderungsanalyse mit der Phase der Sicherheitsanalyse endet. Am Ende dieser Phase sind alle Benutzerrechte prädikativ zu formalisieren.

Mit dem vorgestellten Prozess zur Sicherheitsanalyse und dessen Eingliederung in den Softwarelebenszyklus werden alle ermittelten Anforderungen an den Prozess erfüllt. Die vorgestellte Methode unterstützt in allen Phasen der Anforderungsdefinition eine durchgängige und sukzessive Entwicklung der Sicherheitsaspekte. Sicherheitsaspekte werden innerhalb jedes Prozessabschnitts ermittelt und die Modelle werden schrittweise in sich verfeinert oder in neue, verfeinerte Modelle überführt.

Bei der Modellierung der funktionalen Anforderungen werden stets die Anforderungen an die Zugriffssicherheit mitbetrachtet. Sicherheits- und Funktionsanforderungen werden so gemeinsam entwickelt. Bei der Analyse des Anwendungskerns und der Abläufe werden die Schutzziele und Benutzerrechte modelliert, bei der Modellierung der Anwendungsfälle Sicherheitsanwendungsfälle eingefügt und dabei die strukturelle Beschreibung ergänzt. Das Klassendiagramm der Analyse wurde zur Umsetzung der Sicherheitsaspekte um Vorgangs- und Fachklassen ergänzt und in die Szenarien wurden die Sicherheitsaspekte funktional interpretiert. Durch das vorgestellte Verfahren auch die Prozessanforderung der lokalen Spezifikation, d.h. die Spezifikation von Aspekten der Zugriffssicherheit mit den funktionalen Anforderungen, erfüllt, da die Sicherheitsaspekte in den verschiedenen Abschnitten in kleinen Einheiten ermittelt und dokumentiert werden. Es muss also nicht abschließend ein eigenständiger Prozess zur Modellierung der Sicherheit ablaufen. Die iterative Entwicklung wird durch die zahlreich aufgezeigten Iterationsmöglichkeiten unterstützt.

Anhand der durchgängigen Fallstudie, dem Projektverwaltungssystem *TimeTool*, wurden die erarbeiteten Konzepte illustriert und deren Anwendbarkeit demonstriert.

Die in dieser Arbeit vorgestellte akteurzentrierte Spezifikation von Benutzerrechten wurde bereits in Fallstudien für ein Gesundheitsinformationssystem und für eine E-Government-Lösung erfolgreich angewandt. Der Ansatz wird in mehreren Richtungen weiterentwickelt. Erstens wird an der Universität Innsbruck an einer Werkzeugunterstützung zum vorgestellten Konzept geforscht, sodass Methodenberechtigungen, Akteurspezifikationen und Kategorien mit Kategorieberechtigungen innerhalb eines UML-Tools zur Verfügung stehen. Weiterhin wird der Spezifikationsansatz für den Einsatz in implementierungsorientierten Kontexten erweitert. Im Projekt SECTINO an der Universität Innsbruck findet der akteurzentrierte Ansatz Anwendung bei der Spezifikation von Workflows zwischen Unternehmen. Die Workflows basieren dabei auf Web-Services. Ferner werden Rechte in kombinierten Systemen getestet und analysiert.

Um den Gewinn der vorgestellten Methode quantifizieren zu können, sind Untersuchungen durchzuführen, aus denen die Kostenersparnis der vorgestellten Methode ermittelt werden kann. Die Untersuchungen können sowohl an Fallstudien sowie an realen Projekten durchgeführt werden.

Die vorgestellte Methode zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Systeme wurde im Rahmen des Projekts MEwaDis auf die dienstbasierte Entwicklung angewendet [DGP+04b, DGP+04a]. Anstelle der Geschäftsprozesse werden dabei die Dienste mit ihren in Beziehung stehenden Diensten in einer so genannten *logischen Servicearchitektur* dargestellt. Da kein globales Datenmodell zur Verfügung steht, entfällt die

Modellierung der Schutzziele am Anwendungskern. Bei der Beschreibung der Dienste wird dabei die erweiterte strukturelle Anwendungsfallbeschreibung und die Erstellung von Dienstszenarien verwendet. In [Deu05] wird die Verfeinerung der Beziehungen zwischen Diensten untereinander bei der Anforderungsanalyse untersucht. In diesem Zusammenhang ist in Zukunft die durchgängige und sukzessive Entwicklung von Aspekten der Zugriffssicherheit zu integrieren und die Verfeinerung der Produktartefakte aufzuzeigen.

Parallel zur Entwicklung der vorliegenden Arbeit wurde das V-Modell 97 [DW99] überarbeitet. Das neue V-Modell^{TM}XT beschreibt ein flexibles Vorgehensmodell zum Planen und Durchführen von Systemprojekten. Es definiert die Aktivitäten (Tätigkeiten) und Produkte (Ergebnisse), die während der Entwicklung von Systemen durchzuführen beziehungsweise zu erstellen sind. Darüber hinaus legt das V-Modell die Verantwortlichkeiten jedes Projektbeteiligten fest [WEI04]. Da das V-Modell^{TM}XT ein sehr generisches Produktmodell besitzt und bereits Produkte zur Spezifikation von sicheren Systemen integriert sind, ist zu untersuchen, wie sich die Produkte und Prozessbausteine der vorgestellten Methode in die Struktur des V-Modells^{TM}XT integrieren lassen.

Die vorliegende Arbeit konzentriert sich auf die frühen Phasen der Entwicklung zugriffssicherer Systeme. In [Wim05] wird die Entwicklung von Sicherheitsaspekten in den folgenden Phasen untersucht, insbesondere die modellbasierte Entwicklung innerhalb der Design- und der Testphase.

Neben der Zugriffssicherheit gibt es noch eine Reihe weiterer nichtfunktionaler Anforderungen, wie zum Beispiel Zuverlässigkeit, Effizienz, Benutzbarkeit, Änderbarkeit und Übertragbarkeit. Im Gegensatz zu diesen weiteren nichtfunktionalen Anforderungen kann die Zugriffssicherheit nach Schutzzielen weiter unterteilt werden, sodass eine strukturelle Ermittlung von Anforderungen und schrittweise Umsetzung möglich ist. Für eine Ausweitung der methodischen Entwicklung von nichtfunktionalen Anforderungen ist zu untersuchen, inwiefern die weiteren nichtfunktionalen Anforderungen ebenfalls unterteilt werden können. Erste Ideen und Fragestellungen werden in [FH97b] diskutiert.

Literaturverzeichnis

[Alt04] Ewgeney Alter. Werkzeugunterstützte Analyse von Geschäftsprozessen. Diplomarbeit, Institut für Informatik der Technischen Universität München, 2004.

[Amb98] Scott W. Ambler. *Process Patterns: Building Large-Scale Systems Using Object Technology*. Cambridge University Press, 1998.

[Amb99] Scott W. Ambler. *More Process Patterns: Delivering Large-Scale Systems Using Object Technology*. Cambridge University Press, 1999.

[And01] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Computer Publishing, 2001.

[AW03a] Khaled Alghathbar and Duminda Wijesekera. AuthUML: A Three-phased Framework to model Secure Use Cases. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS) 2003, October 27-31, Washington, DC, U.S.A.* ACM, 2003. `http://www.acm.org/sigs/sigsac/ccs/CCS2003/`.

[AW03b] Christine Artelsmair and Roland R. Wagner. Towards a Security Engineering Process. In Nagib Callaos, William Lesso, Belkis Sánchez, and Elizabeth Hansen, editors, *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, July 27-30 2003*, volume VI of *IS, Technologies and Applications I*, 2003.

[Bal96] Helmut Balzert. *Lehrbuch der Software-Technik: Software-Entwicklung*. Spektrum Akademischer Verlag, first edition, 1996.

[Bal98] Helmut Balzert. *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, first edition, 1998.

[Bas93] Richard Baskerville. Information Systems Security Design Methods – Implications for Information Systems Development. *ACM Computing Surveys*, 25(4):375–414, December 1993.

[Bau97] Friedrich L. Bauer. *Entzifferte Geheimnisse: Methoden und Maximen der Kryptologie*. Springer-Verlag, second edition, 1997.

[BBEH03] Ruth Breu, Klaus Burger, Martin Eberle, and Michael Hafner. *Projekthandbuch zum Projekt TimeTool im Sommersemester 2003*. Universität Innsbruck, März 2003. `http://qe-informatik.uibk.ac.at/lehre/ss03/se4/Projektdurchfuehrung/Projekthandbuch.pdf`.

[BBH+03] Ruth Breu, Klaus Burger, Michael Hafner, Jan Jürjens, Gerhard Popp, Guido Wimmel, and Volkmar Lotz. Key Issues of a Formally Based Process Model for Security Engineering. In *Proceedings of the 16th International Conference on Software & Systems Engineering and their Applications (ICSSEA03), Paris, December 2 - 4, 2003*, 2003.

[BBHP04] Ruth Breu, Klaus Burger, Michael Hafner, and Gerhard Popp. Towards a Systematic Development of Secure Systems. *Information Systems Security*, 13(3):5–13, July/August 2004.

[BBHP05] Ruth Breu, Klaus Burger, Michael Hafner, and Gerhard Popp. Core Concepts of a Process Model for Security Engineering. *SoSyM – Journal on Software & System Modeling*, 2005. To appear.

[BDL03] David A. Basin, Jürgen Doser, and Torsten Lodderstedt. Model Driven Security for Process-Oriented Systems. In *8th ACM Symposium on Access Control Models and Technologies*. ACM, 2003.

[BKL01] Gerald Brose, Manuel Koch, and Klaus-Peter Löhr. Integrating Security Policy Design into the Software Development Process. Technical Report B-01-06, Freie Universität Berlin, November 2001.

[BL73] D. Elliot Bell and Leonard J. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report MTR 2547, Vol 2, MITRE Corp, Bedford, MA, November 1973.

[BN89] David F.C. Brewer and Michael J. Nash. The Chinese Wall Security Policy. In *Proceedings of the 1989th IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[Boe86] Barry Boehm. A Spiral Model of Software Development and Enhancement. In *ACM Sigsoft Software Engineering Notes, Vol. 11, No. 4*, 1986.

[Boo94] Grady Booch. *Object-Oriented Analysis and Design – with Applications*. Addison Wesley Longman, Inc., second edition, 1994.

[BP04] Ruth Breu and Gerhard Popp. Actor-Centric Modeling of User Rights. In T. Margaria-Steffen M. Wermelinger, editor, *Proceedings FASE 2004, Barcelona, March 29-31*, volume 2984 of *LNCS*, pages 165–179. Springer-Verlag, 2004.

[Bre01] Ruth Breu. *Objektorientierter Softwareentwurf – Integration mit UML*. Springer-Verlag, 2001.

[Bre04] Ruth Breu. An Integrated Approach to Use Case Based Development, 2004. To appear.

[BRJ98] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley Longman Inc., first edition, 1998.

[Bro98a] Manfred Broy. *Informatik: Eine grundlegende Einführung, Band 1: Programmierung und Rechnerstrukturen*. Springer-Verlag, second edition, 1998.

[Bro98b] Manfred Broy. *Informatik: Eine grundlegende Einführung, Band 2: Systemstrukturen und Theoretische Informatik*. Springer-Verlag, second edition, 1998.

[BSI99] BSI. Common Criteria for Information Technology Security Evaluation, Version 2.1. Technical report, Bundesamt für Sicherheit in der Informationstechnik, August 1999. `http://www.commoncriteria.org/docs/index.html`.

[BSW01] Imran Bashir, Enrico Serafini, and Kevin Wall. Securing Network Software Applications. *Communications of the ACM*, 44(2):29–30, February 2001.

[CAB+94] Derek Coleman, Patrik Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, and Paul Jeremaes. *Object-Oriented Development: The Fusion Method*. Prentice-Hall, 1994.

[CY91a] Peter Coad and Edward Yourdon. *Object-Oriented Analysis*. Yourdon Press, Englewood Cliffs, second edition, 1991.

[CY91b] Peter Coad and Edward Yourdon. *Object-Oriented Design*. Yourdon Press, Englewood Cliffs, 1991.

[Den91] Ernst Denert. *Software-Engineering*. Springer-Verlag, 1991.

[Deu05] Martin Deubler. *Dienst-orientierte Softwaresysteme: Anforderungen und Entwurf*. PhD thesis, Technische Universität München, 2005. To appear.

[DGP+04a] Martin Deubler, Johannes Grünbauer, Gerhard Popp, Guido Wimmel, and Christian Salzmann. Tool Supported Development of Service Based Systems. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC) 2004, Busan (Korea), November 30 - December 3*, pages 99–108. IEEE Computer Society, 2004.

[DGP+04b] Martin Deubler, Johannes Grünbauer, Gerhard Popp, Guido Wimmel, and Christian Salzmann. Towards a Model-Based and Incremental Development Process for Service-Based Systems. In M. H. Hamaza, editor, *Proceedings of the IASTED International Conference on Software Engineering (IASTED SE) 2004, Innsbruck (Austria), Februar 17-19*, pages 183–188, 2004.

[Dos01] Shreyas Doshi. Software Engineering and Security: Towards Architecting Secure Software. Term Paper for ICS 121, University of California, 2001.

[DW98] Desmond F. D'Souza and Alan C. Wills. *Objects, Components, and Frameworks With UML: The Catalysis Approach*. Addison Wesley Publishing Company, 1998.

[DW99] Wolfgang Dröschel and Manuela Wiemers, editors. *Das V-Modell 97*. Oldenbourg Wissenschaftsverlag GmbH, 1999.

[Eck03] Claudia Eckert. *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, second edition, 2003.

[EP00] Hans-Erik Eriksson and Magnus Penker. *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, Inc., 2000.

[FCK03] David F. Ferraiolo, Ramaswamy Chandramouli, and D. Richard Kuhn. *Role-Based Access Control*. Artech House Publishers, first edition, April 2003.

[FH97a] E. B. Fernandez and J. C. Hawkins. Determining Role Rights from Use Cases. In *Workshop on Role-Based Access Control*, pages 121–125. ACM, 1997.

[FH97b] E. B. Fernandez and J. C. Hawkins. Extending Use Cases and Interaction Diagrams to Develop System Architecture Requirements. Technical Report TR-CSE-97-47, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida, 1997.

[Fir03] Donald G. Firesmith. Security Use Cases. *Journal of Object Technology*, 2(3):53–64, May-June 2003. `http://www.jot.fm/issues/issue_2003_05/column6`.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. *The SSL Protocol – Version 3.0*. Netscape Communications, 1996. `http://wp.netscape.com/eng/ssl3/ssl-toc.html`.

[Fla97] David Flanagan. *Java in a Nutshell – A Desktop Quick Reference*. O'Reilly & Associates, Inc., second edition, May 1997.

[FS98] Martin Fowler and Kendal Scott. *UML Distilled – Applying the Standard Object Modelling Language*. Addison Wesley Longman Inc., 7th edition, June 1998.

[FSG+01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role-Based Access Control. In *ACM Transactions on Information and System Security*, number 3, pages 224–274. ACM, August 2001. http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf.

[GDB] GDBA. SEC: Anwendung des V-Modells und der ITSEC. In *AU250: Vorgehensmodell „V-Modell", Teil 3 Handbuchsammlung*. `http://www.informatik.uni-bremen.de/uniform/gdpa/part3_d/p3sec.htm`.

[GHJV98] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, Inc., 15th edition, 1998.

[GHS03] Tim Grance, Joan Hash, and Marc Stevens. Security Considerations in the Information System Development Life Cycle. Special Publication 800-64, National Institute of Standars and Technology (NIST) Technology Administration U.S. Department of Commerce, October 2003. `http://downloads.securityfocus.com/library/NIST-SP800-64.pdf`.

[GMP+01a] Michael Gnatz, Frank Marschall, Gerhard Popp, Andreas Rausch, and Wolfgang Schwerin. Modular Process Patterns supporting an Evolutionary Software Development Process. In *Proceedings of the Third International Conference on Product Focused Software Process Improvement*, 2001.

[GMP+01b] Michael Gnatz, Frank Marschall, Gerhard Popp, Andreas Rausch, and Wolfgang Schwerin. Towards a Living Software Development Process based on Process Patterns. In V. Ambriola, editor, *Proceedings of the Eight European Workshop*

on Software Process Technology 2001, number 2077 in Lecture Notes in Computer Science, pages 182–202. Springer-Verlag, 2001.

[GMP+02a] Michael Gnatz, Frank Marschall, Gerhard Popp, Andreas Rausch, Maura Rodenberg-Ruiz, and Wolfgang Schwerin, editors. *Proceedings of the 1st Workshop on Software Development Process Patterns at the 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2002), Seattle, Washington, USA, November 4-8, 2002*, Technical Report TUM-I0213. Technische Universität München, December 2002.

[GMP+02b] Michael Gnatz, Frank Marschall, Gerhard Popp, Andreas Rausch, and Wolfgang Schwerin. Towards a Tool Support for a Living Software Development Process. In *HICSS 33, Proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, January 2002.

[GMP+03a] Michael Gnatz, Frank Marschall, Gerhard Popp, Andreas Rausch, and Wolfgang Schwerin. Enabling a Living Software Development Process with Process Patterns. Technical Report TUM-I0310, Technische Universität München, July 2003.

[GMP+03b] Michael Gnatz, Frank Marschall, Gerhard Popp, Andreas Rausch, and Wolfgang Schwerin. The Living Software Development Process. *Software Quality Professional*, 5(3):4–16, June 2003.

[GSG99] Stefanos Gritzalis, Diomidis Spinellis, and Panagiotis Georgiadis. Security Protocols over Open Networks and Distributed Systems: Formal Methods for their Analysis, Design, and Verification. *Computer Communications*, 22(8):695–707, 1999.

[HB03] Michael Howard and David Le Blanc. *Writing Secure Code*. Microsoft Press, second edition, 2003.

[HJ03] Sebastian Höhn and Jan Jürjens. Automated Checking of SAP Security Permissions. In *Proceedings of the 6th IFIP WG 11.5 Working Conference on Integrity and Internal Control in Information Systems (IICIS), Nov. 13-15, 2003, Lausanne, Switzerland*. Kluwer, 2003.

[HNS00] Christine Hofmeister, Robert Nord, and Dilip Soni. *Applied Software Architecture*. Addison Wesley Longman, Inc., 2000.

[HS93] Erika Horn and Wolfgang Schubert. *Objektorientierte Software-Konstruktion: Grundlagen, Modelle, Methoden, Beispiele*. Carl Hanser Verlag, 1993.

[IAB] IABG. *V-Modell*. `http://www.v-modell.iabg.de`.

[IBM03] Business Consulting Services IBM. *SAP Berechtigungswesen, Design und Realisierung von Berechtigungskonzepten für SAP R/3 und SAP Enterprise Portal*. SAP Press, 2003.

[ITS90] ITSEC. Information Technology Security Evaluation Criteria – Harmonised Criteria of France, Germany, the Netherlands, the United Kingdom, May 1990.

[Jac87] Ivar Jacobson. Object-oriented development in an industrial environment. In *Proceedings of OOPSLA'87*, Special issue of SIGPLAN Notices 22(12), pages 183–191, December 1987.

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process.* Addison Wesley Longman, Inc., first edition, 1999.

[JCJO92] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard. *Object-Oriented Software Engineering: A Use-Case Driven Approach.* Reading, MA: Addison-Wesley, 1992.

[JPW03] Jan Jürjens, Gerhard Popp, and Guido Wimmel. Use Case Oriented Development of Security-Critical Systems. *Information Security Bulletion*, 8(2):55–60, March 2003.

[Jür04] Jan Jürjens. *Secure Systems Development with UML.* Springer-Verlag, first edition, 2004.

[JW03] Jan Jürjens and Guido Wimmel. Security Modelling for Electronic Commerce: The Common Electronic Purse Specifications. In *Proceedings of the First IFIP Conference on E-Commerce, E-Business and E-Government (I3E) 2003.* Kluwer, 2003.

[KE01] Alfons Kemper and André Eickler. *Datenbanksysteme – eine Einführung.* Oldenbourg Verlag, 4th edition, 2001.

[KPP03] Manuel Koch and Francesco Parisi-Presicce. UML Specification of Access Control Policies and their Formal Verification. Submitted, 2003.

[Kro97] Petr Kroha. *Softwaretechnologie.* Prentice Hall Verlag GmbH, 1997.

[Kru00] Philippe Kruchten. *The Rational Unified Process: An Introduction, Second Edition.* Addison Wesley Longman, Inc., 2000.

[LBD02] Torsten Lodderstedt, David A. Basin, and Jürgen Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook, editors, *UML 2002 - The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools. 5th International Conference, Dresden, Germany, September/October 2002, Proceedings*, volume 2460 of *LNCS*, pages 426–441. Springer-Verlag, 2002.

[MFSK97] John A. Miller, Mei Fan, Amit P. Sheth, and Krys J. Kochut. Security in Web-Based Workflow Management Systems. In *Proceedings of the International Workshop on Research Directions in Process Technology*, Nancy, France, July 1997.

[MFW+99] John A. Miller, Mei Fan, Shengli Wu, Ismailcem B. Arpinar, Amit P. Sheth, and Krys J. Kochut. Security for the METEOR Workflow Management System. Technical Report UGA-CS-LSDIS-TR-99-010, University of Georgia, June 1999.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Inc., 1996.

[Oak01] Scott Oaks. *Java Security*. O'Reilly & Associates, Inc., second edition, 2001.

[OHJ+99] Bernd Oestereich, Peter Hruschka, Nicolai Josuttis, Hartmut Kocher, Hartmut Krasemann, and Markus Reinhold. *Erfolgreich mit Objektorientierung*. Oldenbourg Wissenschaftsverlag GmbH, 1999.

[OMG03] OMG. Unified Modeling Language Specification – Version 1.5, March 2003.

[OW96] Thomas Ottmann and Peter Widmayer. *Algorithmen und Datenstrukturen*. Spektrum Akademischer Verlag, 3. edition, 1996.

[Par98] Helmuth Partsch. *Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme*. Springer-Verlag, 1998.

[PB96] Gustav Pomberger and Günther Blaschek. *Software-Engineering: Prototyping und objektorientierte Software-Entwicklung*. Carl Hanser Verlag, zweite edition, 1996.

[PBJW03] Gerhard Popp, Ruth Breu, Jan Jürjens, and Guido Wimmel. Security-Critical System Development with Extended Use Cases. In *Proceedings of the 10th Asia-Pacific Software Engineering Conference (APSEC) 2003, Chiang Mai (Thailand), December 10-12*, pages 478–487. IEEE Computer Society, 2003.

[Rat00] Rational Software Corporation. *Rational Unified Process, HTML Documentation*, 2000. Version 2000.02.10.

[Rau01] Andreas Rausch. *Componentware: Methodik des evolutionären Architekturentwurfs*. PhD thesis, Technische Universität München, 2001.

[RBP+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall Verlag GmbH, 1991.

[RE99] Wolfgang Rankl and Wolfgang Effing. *Handbuch der Chipkarten: Aufbau – Funktionsweise – Einsatz von Smart Cards*. Carl Hanser Verlag, third edition, 1999.

[RN03] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Pearson Education, Inc., second edition, 2003.

[Roy70] Walker W. Royce. Managing the Development of Large Software Systems. *Proceedings IEEE WESTCON, Los Angeles*, 14:1–9, August 1970. Reprinted in *Proceedings of the Ninth International Conference on Software Engineering*, March 1987, pp. 328–338.

[RP97] Peter Rechenberg and Gustav Pomberger, editors. *Informatik-Handbuch*. Carl Hanser Verlag, 1997.

[Rup02] Josef Rupprecht. Datensicherheit im Data Warehousing. Technical Report BE HSG/CC DW2/06, Universität St. Gallen, September 2002.

[San96] Ravi Sandhu. Role Hierarchies and Constraints for Lattice-Based Access Controls. In *Proceedings of the European Symposium on Research in Security and Privacy*, 1996.

[Sch92] August-Wilhelm Scheer. *Architektur integrierter Informationssysteme: Grundlagen der Unternehmensmodellierung.* Springer-Verlag, second edition, 1992.

[Sch99] August-Wilhelm Scheer. *ARIS, Vom Geschäftsprozeß zum Anwendungssystem.* Springer-Verlag, 1999.

[Sch00] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World.* John Wiley & Sons, Inc., 2000.

[Sil04] Silicon.de. BKA verzeichnet Anstieg der Computerkriminalität – Aufklärungsquote sinkt. Newsletter vom 04. Mai, 2004. `http://www.silicon.de/cpo/news-itsecurity/detail.php?nr=14478&kategorie=news-itsecurity`.

[SO01] Guttorm Sindre and Andreas L. Opdahl. Capturing Security Requirements through Misuse Cases. In *Proceedings of the Norsk Informatikkonferanse (NIK) 2001*, 2001. `http://www.nik.no/2001`.

[Som00] Ian Sommerville. *Software Engineering.* Addison Wesley Longman, Inc., 2000.

[SS75] Jerome H. Saltzer and Michael D. Schroeder. Protection of Information in Computer Systems. In *Proceedings of the IEEE*, volume 63, pages 1278–1308, March 1975. `http://www.cs.virginia.edu/~evans/cs551/saltzer/`.

[Sta01] Josef Staud. *Geschäftsprozessanalyse: Ereignissgesteuerte Prozessketten und objektorientierte Geschäftsprozessmodellierung für Betriebwirtschaftliche Standardsoftware.* Springer-Verlag, zweite edition, 2001.

[Thu04] Veronika Thurner. *Formal fundierte Modellierung von Geschäftsprozessen – Geschäftsprozesse anschaulich und präzise dokumentieren.* Logos Verlag, 2004.

[Vet01] Monika Vetterling. Security Engineering nach den Common Criteria – Eine Fallstudie. Master's thesis, Technische Universität München, August 2001.

[VM02] John Viega and Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison-Wesley, 2002.

[vOL02] David von Oheimb and Volkmar Lotz. Formal Security Analysis with Interacting State Machines. In D. Gollmann, G. Karjoth, and Waidner M., editors, *Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS) 2002*, number 2502 in Lecture Notes in Computer Science, pages 212–228. Springer-Verlag, 2002.

[VWW02] Monika Vetterling, Guido Wimmel, and Alexander Wißpeintner. Secure Systems Development Based on the Common Criteria. In *10th International Symposium on the Foundations of Software Engineering (FSE-10)*, 2002.

[WEI04] Web-Seite des neuen V-Modell^{TM} XT - Der Entwicklungsstandard für IT-Systeme des Bundes, 2004. `http://www.v-modell-xt.de/`.

[Wik] Wikimedia Foundation Inc. *WIKIPEDIA – Die freie Enzyklopädie im Web.* `http://www.wikipedia.org/`.

[Wim05] Guido Wimmel. *Model-Based Development of Security Critical Systems.* PhD thesis, Technische Universität München, 2005. To Appear.

[WK99] Jos Warmer and Anneke G. Kleppe. *The Object Constraint Language – Precise Modeling with UML.* Addison Wesley Longman, Inc., first edition, 1999.

Tabellenverzeichnis

Abbildungsverzeichnis